

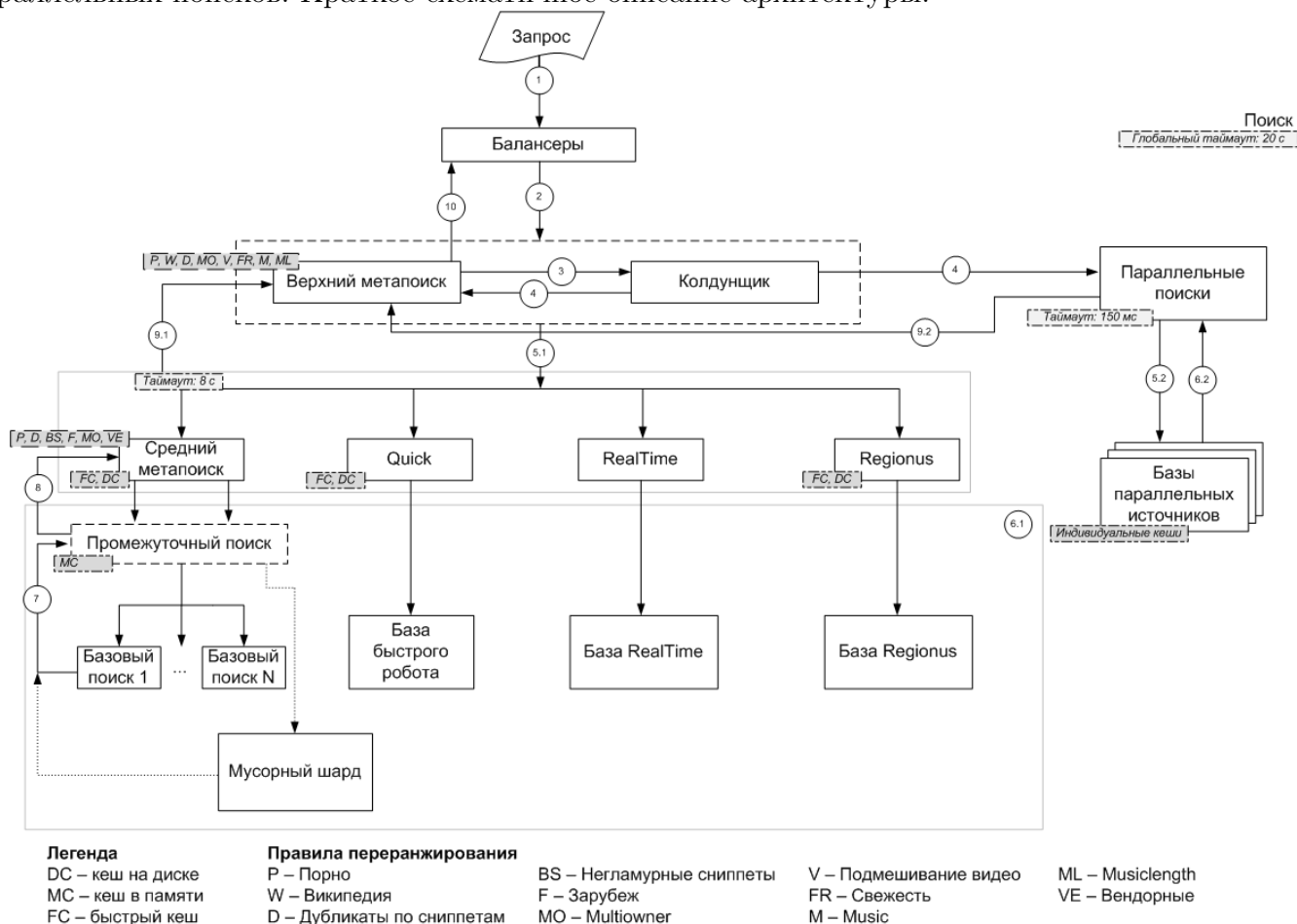
Разработка алгоритма для выделения частовстречающихся шаблонов ошибок из log-файлов программ

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 Подготовка репозитория	6
1.4.2 Выбор хранилища для паттернов	7
1.4.3 Написание программного кода	7
2 Имплементация задачи	8
2.1 Диаграмма компонентов	8
2.2 Описание способов запуска	8
2.3 Анализ полученных результатов	8
3 Дальнейшее развитие	9
Заключение	10

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, колдунчики, антироботы, свежесть, региональные поиск, несколько десятков параллельных поисков. Краткое схематичное описание архитектуры:



Всего одновременно 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-

¹инстанс — 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 экземпляров приложений на отдельных host'ах, что, в свою очередь, замедляет скорость реакции на непредвиденную ситуацию. Но даже автоматизация процесса анализа log-файла на одного экземпляра не решает проблему полностью, поэтому необходима возможность быстрого анализа log-файлов сразу множества экземпляров.

Таким образом, целью этой учебно-исследовательской работы является разработка алгоритма, позволяющего собирать статистику по ошибкам, встречающимся в log-файлах экземпляров 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 MapReduce¹ 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(паттерн², 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 Yandex Tables. 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, имеющего клиент-серверную архитектуру и позволяющего отсоединиться от текущей сессии, оставляя её работать в фоновом режиме с последующей возможностью перепоключения. 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 исполняться

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

²паттерн — регулярное выражение

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.

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

- Для управления версиями исходных кодов использовалась децентрализованная система управления версиями git, в качестве hosting-провайдера был использован сервис github

В качестве языка программирования был выбран python2.7. Так как:

- Он предустановлен в большинстве современных дистрибутивах операционных систем.
- Выразителен. Аналогичные программы на таких языках как Java, C++ имеют в разы большие объёмы исходных кодов.
- Обладает высокой производительностью.
- Имеет множество библиотек, в том числе библиотеки для работы с регулярными выражениями, обёртки для MapReduce.

1.3 Структура данных

В качестве входных данных использовался агрегированный log-файл с нерегулярной структурой, содержащий сообщения об ошибках в различных форматах, как многострочные, так и однострочные.

Первая часть алгоритма, позволяет осуществить сбор статистики и возвращает список пар (паттерн, количество совпадений), так же существует возможность получить часть текста не удовлетворяющую известным паттернам, для последующего анализа с помощью второй части алгоритма, позволяющей выделить новые предполагаемые паттерны и с помощью первой части алгоритма получить статистику, подтверждающую или опровергающую предположение.

1.4 Стадии выполнения задачи

1.4.1 Подготовка репозитория

Был создан git-репозиторий на сервисе github, сделана его локальная копия. Была выбрана следующая структура проекта и правила именования файлов:

- В корневом каталоге лежат файлы README.md, LICENSE, .gitignore.
- В каталоге doc/ хранится документация. Исходные тексты в \LaTeX и скомпилированная версия в PDF.
- В каталоге direlog/ хранятся исходные коды с расширением .py и тесты, имеющие префикс test_
- В каталоге direlog/example/ хранятся примеры log-файлов и вспомогательные скрипты на языке bash.

1.4.2 Выбор хранилища для паттернов

В качестве хранилища для паттернов было решено использовать обычный файл на языке python, названный `patterns.py` и содержащий в себе два списка паттернов `prepare_patterns` и `main_patterns`, используемых на подготовительном этапе и на этапе сбора статистики соответственно, и хранить его под контролем версий в этом же репозитории. Причин на это несколько: во-первых простота модификации файла с помощью скриптов, во-вторых возможность просмотра истории изменений и откат к предыдущим версиям, в-третьих возможность ручного редактирования.

1.4.3 Написание программного кода

Написание программного кода было разделено на несколько этапов.

- 1 Написание утилиты для предварительной обработки исходного log-файла. Утилита получила название `prepare.py` и позволила подготовить сырой log-файл для последующей обработки, путём замены уникальных токенов, таких как `UUID`, `timestamp`, версии, номера строк, пути, содержащие версии, на строковые константы.
- 2 Формирование `prepare_patterns` на основе ручного анализа log-файлов.
- 3 Написание функциональных тестов для `prepare.py`.
- 4 Реализация алгоритма для сбора статистики с использованием `main_patterns`. Утилита получила название `direlog.py`. И позволяет на выходе получить список пар (паттерн, количество совпадений) или текст, не подходящий под известные паттерны.
- 5 Добавление поддержки буферизации входного потока и поддержки многострочных паттернов.
- 6 Написание функциональных тестов для `direlog.py`.
- 7 Добавление функции, позволяющей запускать алгоритм на MapReduce-кластере.

Глава 2

Имплементация задачи

2.1 Диаграмма компонентов

2.2 Описание способов запуска

2.3 Анализ полученных результатов

Глава 3

Дальнейшее развитие

Заключение