МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

Национальный исследовательский ядерный университет
“МИФИ”

Факультет Кибернетики и информационной безопасности
Кафедра Информационные технологии в социальных системах

К защите допущен
Зам. заведующего кафедрой
_____ /Сергиевский М. В./
“_____” _____ 201__г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к учебно-исследовательской работе
и курсовому проекту
на тему:

РАЗРАБОТКА АЛГОРИТМА ДЛЯ ВЫДЕЛЕНИЯ ЧАСТОВСТРЕЧАЮЩИХСЯ ШАБЛОНОВ ОШИБОК ИЗ ФАЙЛОВ ЖУРНАЛОВ ПРИЛОЖЕНИЙ

Студент: Тропин А. Г.

Руководитель Сергиевский М. В.

Оценка

Подпись членов комиссии

Москва 2015

РЕФЕРАТ

Пояснительная записка к учебно-исследовательской работе содержит 45 страниц, 843 строки исходного кода, находящегося в откром доступе, 2 рисунка, 5 приложений.

Ключевые слова: mapreduce, logs, распределенные вычисления, python, надежность, отказоустойчивость, функциональное программирование, регулярные выражения.

Цель работы — разработка алгоритма для выделения частовстречающихся шаблонов ошибок из файлов журналов приложений для повышения скорости реакции на непредвиденные ситуации и повышения стабильности работы распределенной системы.

В процессе работы осуществлялась разработка алгоритма, реализация алгоритма на языке python. И внедрение в производство с использованием MapRedcue-системы Yandex Tables.

В результате работы был разработан алгоритм, реализован на языке python и написаны вспомогательная утилита на языке bash и две утилиты на языке python.

СОДЕРЖАНИЕ

НОРМАТИВНЫЕ ССЫЛКИ

В настоящей пояснительной записке к учебно-исследовательской работе использованы ссылки на следующие стандарты.

ГОСТ 7.32-2001 Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления.

ГОСТ 7.9-95 Система стандартов по информации, библиотечному издательскому делу. Реферат и аннотация. Общие требования.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Определение 1 (MapReduce)**.** Модель распределённых вычислений, представленная компанией Google, используемая для параллельных вычислений над очень большими, несколько петабайт, наборами данных в компьютерных кластерах.

Определение 2 (Map)**.** map(f, list) — функция высшего порядка двух аргументов, применяет к каждому элементу списка list, функцию f(x), в качестве результата возвращает список полученных значений.

Пример. map(lambda x: x**3, [1, 2, 3]) вернёт список [1, 8, 27].

Определение 3 (Reduce)**.** reduce(f, list, init=None) — функция высшего порядка, последовательно применяет f(x, y) к элементу списка и значению от предыдущего выполнения функции.

Пример. reduce(lambda x, y: x + y, [1, 2, 3]) вернёт 6(сумма всех элементов). 6 = sum(sum(1, 2), 3).

Определение 4 (Экземпляр)**.** Приложение, запущенное в контейнере и описываемое парой host:port.

Определение 5 (Шаблон)**.** Специально подготовленное регулярное выбражение с экранированными спецсимволами.

Определение 6 (Эвристический алгоритм (эвристика))**.** Алгоритм решения задачи, не имеющий строгого обоснования, но, тем не менее, дающий приемлемое решение задачи в большинстве практически значимых случаев.

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.

Всего единовременно запущено несколько ******** тысяч экземпляров приложений. Каждый экземпляр генерирует множество сообщений об ошибках и записывает каждое из них в файл журнала ошибок.

Некоторые приложения, близкие по функционалу, пишут в один и тот же файл. Файлы журналов ротируются согласно определённому алгоритму. Тем не менее объем файла журнала для одного экземпляра 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 экземпляров приложений количества узлов на которых может быть запущена программа, реализующая алгоритм.

В организации, в которой выполнялась учебно-исследовательская работа, развёрнута большая 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 файлы журналов отдельных экземпляров приложений на отдельных серверах, что, в свою очередь, замедляет скорость реакции на непредвиденную ситуацию. Но даже автоматизация

процесса анализа файла журнала одного экземпляра не решает проблему полностью, поэтому необходима возможность быстрого анализа файлов журналов сразу множества экземпляров.

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

1 СОВРЕМЕННОЕ СОСТОЯНИЕ ПРОБЛЕМЫ АНАЛИЗА ФАЙЛОВ ЖУРНАЛОВ

1.1 Актуальность задачи

Достаточно крупных систем, для которых поставленная задача могла бы иметь актуальность, не так много и в основном это коммерческие проекты таких компаний как: google, yahoo, yandex, amazon, facebook, twitter. Откуда вытекает два обстоятельства:

а) Продукты разрабатываемые компаниями не выкладываются в свободный доступ.

б) Продукты разрабатываемые сторонними компаниями не учитывают специфику крупных распределённых систем.

Ещё одним важным аспектом является тот факт, что существующие приложения предполагают, что формат файла журнала заранее известен, но это бывает не всегда так. Например, если продукт разрабатывается в течении 20 лет исправление ошибки, заложенной в начале разработки, обходится очень дорого. Поэтому на порядок дешевле написать алгоритм, который позволяет анализировать неструктурированные файлы журналов, чем исправлять архитектурные ошибки.

1.2 Сравнение с существующими аналогами

Ниже представлен список существующих проприетарных и свободных решений, выложенных в открытом доступе, с описанием их недостатков:

– ascolog. https://www.ascolog.com/content/about-ascolog

- Проприетарный. Отсутствие исходного кода создаёт множество проблем, таких как: отсутствие возможности расширения функционала, возможное наличие вредоносных компонентов.
- Необходимая операционная система для запуска: Windows. К сожалению это достаточно редкое решение для крупных распределённых систем. Критичный пункт.
- Нет возможности для горизонтального масштабирования. Вертикальное масштабирование сильно ограниченно по своим возможностям. Ещё один критичный пункт.
- Нет возможности для расшариения функционала, какими-либо встроенными средствами.
- Сайт производителя не обновлялся больше 6 месяцев и, по всей видимости, продукт больше не развивается.

– sawmill. https://www.sawmill.net/index.html

- Проприетарный. Отсутствие исходного кода создаёт множество проблем, таких как: отсутствие возможности расширения функционала, возможное наличие вредоносных компонентов.
- Нет возможности для расшариения функционала, какими-либо встроенными средствами.
- Предполагается заранее известная структура логов. Критичный пункт.

Из-за наличия критичных проблем каждого из вышеперечисленных решений, они не подходят для решения имеющейся проблемы.

1.3 Постановка задачи

Необходимо разработать алгоритм, позволяющий:

– Собирать статистику с использованием известных шаблонов.

– Выделять новые шаблоны с помощью эвристических методов.

– Производить лёгкое масштабирование программы, реализующей алгоритм с использованием существующих технологий.

– Посмотреть контекст сообщения об ошибке(несколько строк сверху и снизу).

Требование к программе, реализующей алгоритм:

– Высокая производительность(файл, размеров в 100МБ должен обрабатываться не более 15 секунд).

– Возможность запуска под операционной системой семейства GNU/Linux.

– Горизонтальная масштабируемость.

2.1 Структура

Структура входных данных(файла журнала) разнородна. Это может быть как стэк вызовов функции языка javascript(несколько строк). Так и ошибки, вызова функций языка perl. Так и отладочные сообщения о входящем запросе неправильного формата. Все эти сообщения имеют принципиально разный формат, к тому же некоторые имеют мета-информацию(временная отметка, важность и т.д.), другие — нет.

За счёт этого, даже сообщения об одной и той же ошибке могут совпадать неполностью(различные временные отметки) или практически полностью несовпадать — информация об ошибке и длинный путь до файла.

Поэтому на первом этапе необходимо унифицировать сообщения об ошибках за счёт замены мета-информации и дополнительных сведений на строковые константы:

Пример (Замена мета-информации и доп. сведений). UUID вида ACE088EB-ECA6-4348-905A-041EF10DBD53 можно заменить на строковую константу вида UUID_SUBSTITUTE.

Пример (Замена мета-информации и доп. сведений). ip-адрес вида 10.205.10.74 можно заменить на строковую константу вида IP_ADDRESS_SUBSTITUTE.

После этого данные станет проще анализировать с помощью алгоритмов, так как одинаковые ошибки будут совпадать, независимо от времени появления или идентификатора запроса.

Для поиска и замены удобно использовать регулярные выражения, например:

– `'(\d{16}-[-\w]*\b)', 'REQUEST_ID_SUBSTITUTE'`

- ```
  '([0-9A-F]){8}-[0-9A-F]{4}-[0-9A-F]{4}-
  [0-9A-F]{4}-[0-9A-F]{12}', 'UUID_SUBSTITUTE'
  ```

- ```
  '\b(\d{1,3}\.){3}\d{1,3}\b', 'IP_ADDRESS_SUBSTITUTE'
  ```

- ```
  '\w{3} \w{3} \d{1,2} \d{1,2}:\d{1,2}:
  \d{1,2} \d{4}', 'TIMESTAMP_SUBSTITUTE'
  ```

- ```
  'line \d+', 'LINE_SUBSTITUTE'
  ```

Такого рода пары(регулярное выражение, строковая константа) удобно хранить и использовать. Регулярные выражения — гибкий инструмент, поддерживаемый практически всеми современными языками программирования.

Сами сообщения об ошибках можно представить в виде регулярных выражений. Но зачем тогда делать первый этап преобразования входных данных? Теоретически можно описать сообщение об ошибке регулярным выражением, но на практике это регулярное выражение будет достаточно сложным и нечитаемым, поэтому подготовка начальных данных сильно упрощает регулярные выражения, используемые для выделения ошибки. Например следующий многострочный шаблон было бы воспринимать человеку ещё сложнее, если бы, вместо строковых констант были бы регулярные выражения:

```
"""JavaScript\.\ TypeError\:\ Object\
\#\<t\>\ has\ no\ method\ \'Ukraine\'
\ \ \ \ at\ Object\.blocks\.b\-counters\
_\_gemius\ \(web3\_exp\/pages\-desktop\/
search\/\_search\.all\.priv\.js\:POSITION\_SUBSTITUTE\)
\ \ \ \ at\ Object\.\[object\ Function\]\
.Object\.toString\.call\.e\.\(anonymous\ function\)\
\[as\ b\-counters\_\_gemius\]\ \(web3\_exp\/pages\
-desktop\/search\/\_search\.all\.priv\.js\:POSITION\_SUBSTITUTE\)
```

```
\ \ \ \ at\ Object\.blocks\.b\-counters\.blocks\.b\-cntrs\
\(web3\_exp\/pages\-desktop\/search\/\_search\.all\
.priv\.js\:POSITION\_SUBSTITUTE\)
\ \ \ \ at\ Object\.\[object\ Function\]\.Object\
.toString\.call\.e\.\(anonymous\ function\)\
\[as\ b\-counters\]\ \(web3\_exp\/pages\-desktop\/
search\/\_search\.all\.priv\.js\:POSITION\_SUBSTITUTE\)
\ \ \ \ at\ Object\.\[object\ Function\]\.Object\.toString\.
call\.e\.\(anonymous\ function\)\ \[as\ b\-page\]\
\(web3\_exp\/pages\-desktop\/search\/\_search\.all\
.priv\.js\:POSITION\_SUBSTITUTE\)\ at\
\/db\/BASE\/upper\-SHARD\_SUBSTITUTE\/arkanavt\/report\
/lib\/YxWeb\/Util\/Template\/JS\.pm\ LINE\_SUBSTITUTE\.\
\(Object\.blocks\.b\-page\ \(web3\_exp\/pages\-desktop\/search\/
\_search\.all\.priv\.js\:POSITION\_SUBSTITUTE\)\)\n\Z""",
```

Также использование дополнительных регулярных выражений в шаблонах, приводит к падению производительности, увеличивает вероятность ошибки при подсчёте статистики и усложняет автоматическое выделение новых шаблонов.

Для подсчёта статистики используется достаточно простая структура — пара (шаблон, количество совпадений). Это один из ключевых моментов, позволяющих легко масштабировать приложение, реализующее алгоритм, с помощью модели MapReduce.

2.2 Архитектура

Для реализации поставленных на этапе анализа требований задачи были выделены следующие элементы:

- Основной модуль содержащий реализацию алгоритма.
- Вспомогательный модуль, позволяющий первичное преобразование данных (замена мета-информации и дополнительной информации на строковые константы).

– Вспомогательная утилита для переноса файлов журналов в распределённое хранилище для последующей обработки.

Для реализации алгоритма были выделены следующие классы:

– Controller — управляющий класс, предназначенный для координации взаимодействия остальных классов.

– Buffer — класс, позволяющий хранить в оперативной памяти некоторую часть файла, для последующей работы с ней других классов.

– StatCollector — класс, объекты которого, хранят пары (шаблон, количество совпадений) и позволяющий осуществлять операции добавления или получения данных.

– Snippet — класс, объекты которого хранят контекст(шаблон и несколько строк в файле журнала, окружающих текст, подходящий под шаблон).

– SnippetQueue — класс, призванный упростить реализацию наполнения объектов класса Snippet. Позволяет централизованно наполнять контекст всех объектов.

UML-диаграмма классов представлена на рисунке 2.1.

2.3 Планируемая выгода

В настоящий момент из-за сложности ручного анализа файл журналов error_log, содержащий большое количество неструктурированной информации и лишних сообщений используется только в крайних случаях, когда другие инструменты не дают достаточно информации о проблеме. Инженер затрачивает от одной до трёх минут на поиск необходимого сообщения с релевантной информацией.

С помощью разрабатываемого алгоритма можно получить следующие улучшения существующего процесса:

– Можно предоставить информацию для разработчиков, позволяющую провести рефакторинг подсистемы вывода сообщений об ошибках,

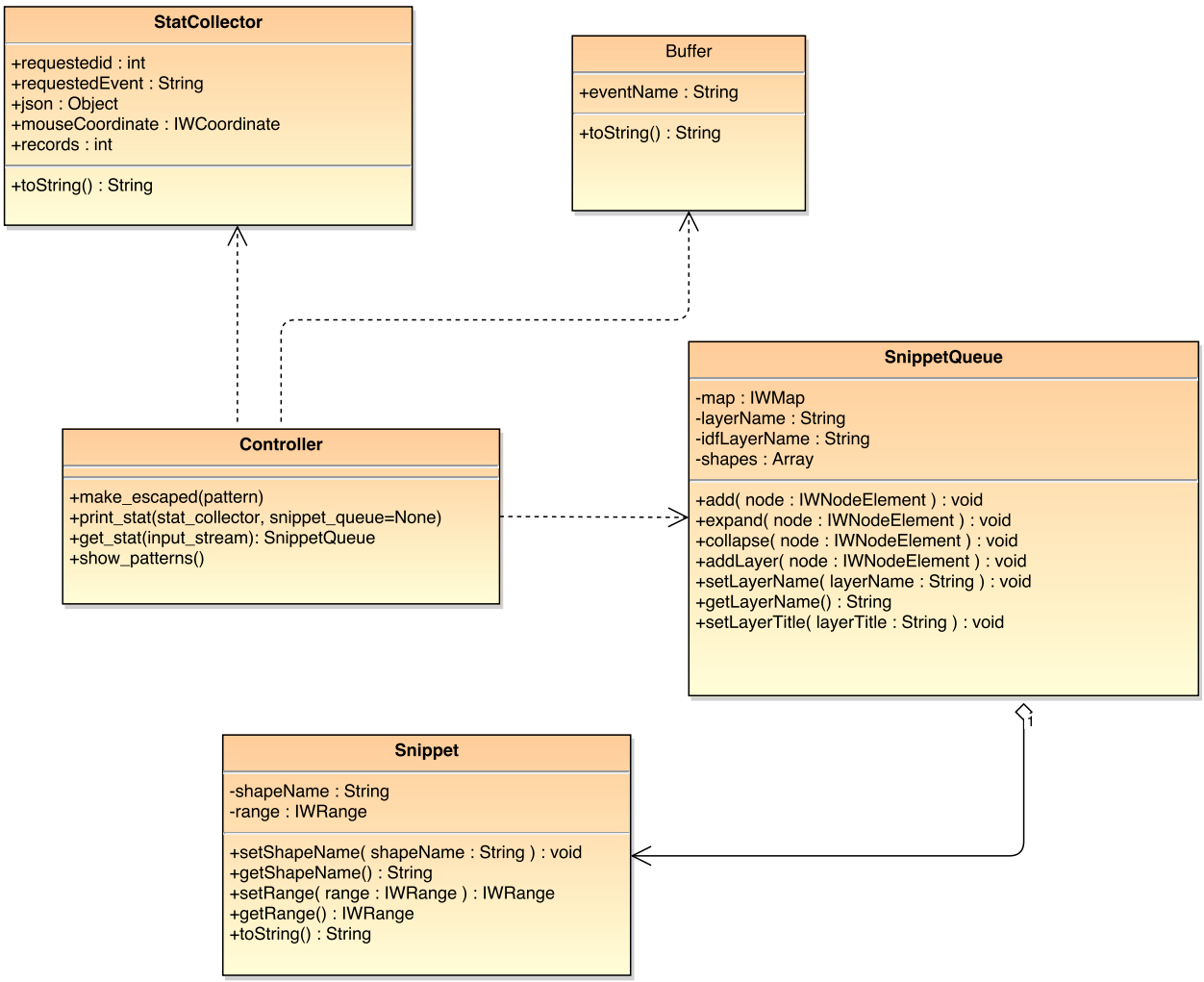

Рис. 2.1: Диаграмма классов

что уменьшит количество неинформативных сообщений и улучшит структуру файла журнала.

– Позволит исключить неинформативные сообщения без изменения существующих приложений из файла журнала и предоставить инженеру максимально информативную выжимку.

– Позволит дать возможность ручного анализа файла журнала инженерам, что уменьшит время диагностики проблем.

– Позволит предсказывать нештатные ситуации за счёт регулярного автоматического анализ.

3 РЕАЛИЗАЦИЯ

3.1 Алгоритм работы программы

3.1.1 Представление задачи в терминах MapReduce

Для параллельного запуска приложений была выбрана модель распределённых вычислений MapReduce по ряду причин. Основные понятия:

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(шаблон, количество совпадений) и благодаря этому однопоточный код легко запускается на множестве узлов MapReduce-кластера. Это обстоятельство освобождает от реализации сложного механизма сетевого взаимодействия.

В качестве реализации модели MapReduce была выбрана реализация Yandex Tables, обладающая рядом отличий и нововведений:

- Колоночное хранение и range-операции
- Дерево метаинформации
- Единая операция Map-Reduce
- Расширенная поддержка транзакций, включая вложенность
- Настраиваемый коэффициент репликации и алгоритмы сжатия данных
- Хранение файлов в системе и их раздача исполняемым задачам
- Разные форматы стриминга (yamr, dsv)
- Надёжность и производительность

- Отказоустойчивый мультимастер
- Отказоустойчивый реплицированный планировщик
- Минорные обновления проходят без заметного эффекта для конечных пользователей

– Гибкие слоты (заказ CPU, RAM)

– Существенные обновления не требуют пересборки C++ клиентов (так как все работает через HTTP API и стриминг клиент)

Не смотря на существенные отличия от модели, описанной в документе от компании Google, код программы, реализующий алгоритм легко переносится на другие реализации данной модели распределённых вычислений.

3.1.2 Структура данных

В качестве входных данных использовался агрегированный файл журнала 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 (шаблон, количество совпадений), так же существует возможность получить часть текста не удовлетворяющую известным шаблонам, для последующего анализа с помощью второй части алгоритма, позволяющей выделить новые предполагаемые шаблоны и с помощью первой части алгоритма получить статистику, подтверждающую или опровергающую предположение.

3.1.3 Алгоритм

На рисунке 3.1 представлена упрощённая схема алгоритма подсчёта количества совпадения шаблонов в терминах модели MapReduce.

На первом этапе(Splitting) происходит разбиение агрегированного файла журнала на части и перемещение частей в оперативную память рабочих узлов MapReduce-кластера.

На втором этапе(Mapping), в рамках каждого рабочего узла, происходит подсчёт количества совпадений каждого шаблона.

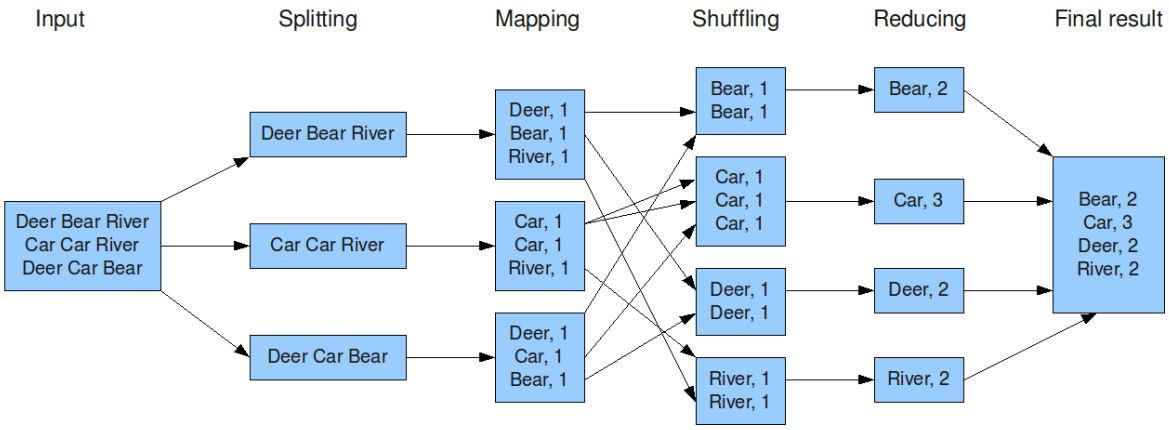

Рис. 3.1: Алгоритм подсчёта количества совпадений шаблонов

На третьем этапе(Shuffling) происходит перераспределение данных между рабочими узлами. Информация про одни и те же шаблоны попадают на один узел.

На четвёртом этапе(Reduce) происходит объединения информации про один шаблон.

На заключительном, пятом этапе, происходит генерация отчёта и запись его на распределённую файловую систему.

3.2 Особенности реализации, проблемы и способы их решения

3.2.1 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 узла. Были выбраны следующие программные продукты:

– В качестве операционной системы использовался дистрибутив

GNU/Linux Ubuntu 12.04 LTS с модифицированным ядром и дополнительными пакетами. Но принципиального отличия при использование приложения на официальном дистрибутиве возникнуть не должно.

– Vim — один из немногих 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 отсоединятся от текущей сессии, оставляя её работать в фоновом режиме с последующей возможностью переподключения. 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 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 сервиса для хранения репозитория 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.

3.2.2 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 LaTeXи скомпилированная версия в PDF.

– В каталоге direlog/ хранятся исходные коды с расширением .py и тесты, имеющие префикс test_

– В каталоге direlog/example/ хранятся примеры файлов журналов и вспомогательные скрипты на языке bash.

3.2.3 Выбор хранилища для шаблонов

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.

3.2.4 Написание программного кода

Для разработки приложения, реализующего алгоритм была использована одна из гибких методологий разработки - экстремальное программирование.

Использовались следующие приёмы этой методологии разработки:

- Короткий цикл обратной связи
  - Разработка через тестирование
  - Заказчик всегда рядом
  - Парное программирование
- Непрерывный, а не пакетный процесс
  - Непрерывная интеграция
  - Рефакторинг
  - Частые небольшие релизы
- Понимание, разделяемое всеми
  - Простота
  - Коллективное владение кодом
  - Стандарт кодирования
- Социальная защищённость программиста
  - 40-часовая рабочая неделя

В результате можно выделить следующие этапы развития приложения:

а) Написание функциональных тестов для prepare.py.

б) Написание утилиты для предварительной обработки исходного файла журнала. Утилита получила название prepare.py и позволила подготовить сырой файл журнала для последующей обработки, путём замены уникальных токенов, таких как UUID, timestamp, версии, номера строк, пути, содержащие версии, на строковые константы.

в) Формирование prepare_patterns на основе ручного анализа файлов журналов.

г) Написание функциональных тестов для direlog.py.

д) Реализация алгоритма для сбора статистики с использованием main_patterns. Утилита получила название direlog.py. И позволяет на выходе получить список пар (шаблон, количество совпадений) или текст, не подходящий под известные шаблон.

е) Написание функциональных тестов для direlog.py.

ж) Добавление поддержки буферизации входного потока и поддержки многострочных шаблонов.

з) Добавление функции, позволяющей запускать алгоритм на MapRedcue-кластере.

Для сбора файлов журналов используется приложение collect-logs.py, представленное в приложении Д, в котором используется коммерческая технология skynet. При использовании данного приложения в среде, где отсутствует вышеупомянутая технология стоит использовать такие альтернативы как rbtorrent или rsync.

ЗАКЛЮЧЕНИЕ

В процессе выполнения учебно-ислледовательской работы был разработан алгоритм, позволяющий собирать статистику количества совпадений шаблонов в файлах журналов. Алгоритм был реализован на языке python с использование MapReduce-системы Yandex Tables.

Кроме сбора статистики приложение позволяет получить контекст шаблона, что полезно в множестве случаев при анализе файла журнала.

С помощью собранной статистики, приложение позволило продемонстритровать проблемы существующей подсистемы отладочного вывода и форсировать её рефакторинг.

После рефакторинга скорость роста файла журнала уменьшилась с полутора мегабайт в секунду до одного мегабайта в секунду. В отладочные сообщения была добавлена дополнительная информация.

Так же разработанное приложение позволяет исключить заведомо неинформативные сообщения об ошибках, что в совокупности с предыдущим фактом уменьшает время ручного анализа файла журнала инженером с двух-трёх минут до тридцати секунд.

Ввиду сложности оценки корректности предположения, не было реализовано автоматическое выделение шаблонов с помощью эвристики. Поэтому выделение новых шаблонов осуществляется в ручном режиме.

Направление дальнейшего развития приложения — автоматическое выделение шаблонов средствами машинного обучения.

ЛИТЕРАТУРА

1. John Bent, Douglas Thain, Andrea C.Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, and Miron Livny. “Explicit control in a batch-aware distributed file system.” In Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation NSDI, March 2004.

2. Douglas Thain, Todd Tannenbaum, and Miron Livny. “Distributed computing in practice: The Condor experience. Concurrency and Computation: Practice and Ex- perience”, 2004.

3. William Gropp, Ewing Lusk, and Anthony Skjellum. “Using MPI: Portable Parallel Programming with the Message-Passing Interface.” MIT Press, Cambridge, MA, 1999.

ПРИЛОЖЕНИЯ

Приложение А

direlog.py

```
#!/usr/bin/env python
# encoding: utf-8

import sys
import os
import re
import argparse
import fileinput
import copy
import itertools

from argparse import RawDescriptionHelpFormatter

from patterns import main_patterns


BUFFER_SIZE = 30

class Buffer(object):
    """Class for buffering input"""

    def __init__(self, output_filename=os.devnull):
        self.buf = [''] * BUFFER_SIZE
        self.line_scrolled = 0
        self.matched_lines = []
        self.unmatched_stream = open(output_filename, 'w')
        self.updated = True
        self.last_text = ''

    def add(self, line):
        self.buf.append(line)
        if 0 not in self.matched_lines:
            self.unmatched_stream.write(line)

        self.buf = self.buf[1:BUFFER_SIZE+1]
```

```
            self.matched_lines = filter(lambda x: x >= 0,
                                        map(lambda x: x - 1, self.matched_lines))
            self.line_scrolled += 1
            self.updated = True

        @property
        def text(self):
            """lazy function, returning joined lines
            :returns: str()

            """
            if self.updated:
                self.last_text = ''.join(self.buf)
                self.updated = False

            return self.last_text

        def try_to_match(self, pattern):
            matched = pattern.search(self.text)
            if matched:
                self.mark_matched(pattern)
            return matched

        def mark_matched(self, pattern):
            lines_count = len(pattern.split('\n'))
            marked_lines = range(BUFFER_SIZE - lines_count, BUFFER_SIZE)
            self.matched_lines += marked_lines

        def __enter__(self):
            return self

        def __exit__(self, type, value, traceback):
            for _ in xrange(BUFFER_SIZE):
                self.add('')
            self.unmatched_stream.close()


    class SnippetsQueue(object):
        def __init__(self):
            self.new_snippets = []
            self.ready_snippets = {}
```

```
        self.pattern_used = {}

    def push(self, snippet):
        if not self.pattern_used.get(pattern, False):
            self.new_snippets.append(snippet)
        self.pattern_used[pattern] = True

    def add(self, line):
        for snippet in self.new_snippets:
            if snippet.full():
                self.make_ready(snippet)
            else:
                snippet.add(line)

    def make_ready(self, snippet):
        self.new_snippets.remove(snippet)
        pattern = snippet.pattern
        try:
            self.ready_snippets[pattern].append(snippet)
        except KeyError:
            self.ready_snippets[pattern] = [snippet]
        self.pattern_used[pattern] = False
        if len(self.ready_snippets[pattern]) == SNIPPETS_TO_SHOW:
            self.pattern_used[pattern] = True

    def make_all_ready(self):
        for snippet in self.new_snippets:
            self.make_ready(snippet)


class StatCollector(object):

    """Class for collecting statistics"""

    def __init__(self):
        self.match_count = {}
        self.lines_count = 0
        self.number_of_matched_lines = 0
        pass

    def add(self, pattern):
```

```
        """Add pattern matching event"""
        if pattern not in self.match_count:
            self.match_count[pattern] = 0
        self.match_count[pattern] += 1
        self.number_of_matched_lines += len(pattern.split('\n'))


def print_stat(stat_collector, snippets_queue=None, **kwargs):

    """Print statistics and snippets"""

    for pattern, count in stat_collector.match_count.iteritems():
        print """\
********************************************************************************
pattern: "{}"
--------------------------------------------------------------------------------
number of matches: {}
********************************************************************************\
""".format(pattern, count)
        if snippets_queue:
            if pattern in snippets_queue.ready_snippets:
                for snippet in snippets_queue.ready_snippets[pattern]:
                    snippet.show()
                    output_stream.write('-' * 80 + '\n')
            else:
                output_stream.write('|No snippets found : (\n')
                output_stream.write('-' * 80 + '\n')

    print 'patterns used: {}/{}'.format(
        len(stat_collector.match_count),
        len(kwargs['patterns']))
    print 'number of lines matched: {}/{}'.format(
        stat_collector.number_of_matched_lines,
        stat_collector.lines_count)



def make_escaped(string):
    """Make escaped pattern from string

    :string: string to be escaped
```

```
    :returns: pattern

    """

    return re.escape(string.replace('\n', 'NSLPH')).replace('NSLPH', '\n') + r'\n\Z'


def get_stat(input_files, snippets_count=0, context=3,
             patterns=main_patterns,
             output_stream=sys.stdout, snippets_file=None,
             unmatched_filename=os.devnull):
    """Show statistics for patterns

    :input_stream: input stream
    :snippets_count: maximum number of snippets to show for every pattern
    :context: number of lines in snippet around last line of pattern match
    :patterns: patterns that need to look
    :output_stream: stream for output
    :returns: None

    """

    input_stream = fileinput.input(input_files)

    LINES_ABOVE = context
    LINES_BELOW = context
    SNIPPETS_TO_SHOW = snippets_count
    SHOW_SNIPPETS = snippets_count > 0

    class Snippet(object):
        def __init__(self, lines_above, pattern, line_number):
            self.text = copy.copy(lines_above)
            self.pattern = pattern
            self.line_number = line_number

        def full(self):
            return len(self.text) >= LINES_ABOVE + LINES_BELOW + 1

        def add(self, line):
            self.text.append(line)
```

```
    def show(self):
        i = 0
        for line in self.text:
            if i == LINES_ABOVE:
                output_stream.write('| {} ==>'.format(self.line_number))
            else:
                if re.search(self.pattern, line):
                    output_stream.write('|-->')
                else:
                    output_stream.write('|>')
            output_stream.write(line)
            i += 1


stat_collector = StatCollector()
snippets_queue = SnippetsQueue()
line_number = 1

compiled_patterns = map(re.compile, patterns)

with Buffer(unmatched_filename) as input_buffer:
    if snippets_file:
        snippets_buffer = Buffer()
    else:
        snippets_buffer = input_buffer

    for line in input_stream:
        input_buffer.add(line)
        if snippets_file:
            line = snippets_file.readline()
            snippets_buffer.add(line)

        for pattern in compiled_patterns:
            if input_buffer.try_to_match(pattern):
                if SHOW_SNIPPETS:
                    if LINES_ABOVE > 0:
                        snippet_begining = snippets_buffer.buf[-LINES_ABOVE:]
                    else:
                        snippet_begining = []

                    snippet = Snippet(snippet_begining, pattern.pattern,
```

```
                                        line_number)
                    snippets_queue.push(snippet)
                stat_collector.add(pattern.pattern)

            if SHOW_SNIPPETS:
                snippets_queue.add(line)
            line_number += 1

    stat_collector.lines_count = line_number - 1

    if not SHOW_SNIPPETS:
        snippets_queue = None
    else:
        snippets_queue.make_all_ready()

    return (stat_collector, snippets_queue)


def show_patterns():
    """Show patterns from patterns.py"""

    for pattern in main_patterns:
        print pattern
        print '-' * 80


def print_escaped(files):
    input_stream = fileinput.input(files)
    text = ''
    for line in input_stream:
        text += line
    escaped_text = make_escaped(text)
    print '\n'
    print escaped_text


def main():
    parser = argparse.ArgumentParser(description=\
    """
        Parse file[s]\n\n
        examples:
```

```
    ./%(prog)s file[s] -s 2 -C 3
    cat error_log | tail -n 1000 | ./prepare.py | ./%(prog)s
    ./%(prog)s -p
    echo "some \\ntext, that wants to be pattern" | ./%(prog)s -e
""", formatter_class=RawDescriptionHelpFormatter)
parser.add_argument('file', nargs='*', default=[],
                    help='file[s] to be parsed')
parser.add_argument('-s', '--snippets', nargs='?', type=int, const=5,
                    help='show maximum SNIPPETS snippets (5 default)')
parser.add_argument('-C', '--context', nargs='?', type=int,
                    help='show CONTEXT lines around pattern last line')
parser.add_argument('-p', '--patterns', action='store_const', const=True,
                    help='show patterns')
parser.add_argument('-e', '--escape', action='store_const', const=True,
                    help='escape given string')
parser.add_argument('-o', '--original', nargs=1,
                    help='provide original file for better snippets')
parser.add_argument('-u', '--unmatched', nargs=1,
                    help='output unmatched text to file')

args = parser.parse_args()

if args.patterns:
    show_patterns()
    return

if args.escape:
    print_escaped(args.file)
    return

kwargs = {}

kwargs['input_files'] = args.file

if args.snippets:
    kwargs['snippets_count'] = args.snippets

if args.context is not None:
    kwargs['context'] = args.context

if args.original:
```

```
        kwargs['snippets_file'] = fileinput.input(args.original)

    if args.unmatched is not None:
        kwargs['unmatched_filename'] = args.unmatched[0]

    kwargs['patterns'] = main_patterns

    result = get_stat(**kwargs)
    print_stat(*result, **kwargs)


if __name__ == '__main__':
    main()
```

Приложение Б

prepare.py

```
#!/usr/bin/env python
# encoding: utf-8
import sys
import re
import argparse
import fileinput

from argparse import RawDescriptionHelpFormatter

from patterns import pre_patterns


def prepare(input_stream, outfile=sys.stdout):
    """
    Apply pre_patterns from patterns to input_stream

    :input_stream: input stream

    """
    compiled_patterns = []
    for pattern in pre_patterns:
        compiled_patterns.append((re.compile(pattern[0]), pattern[1]))

    try:
        for line in input_stream:
            result = line
            for pattern in compiled_patterns:
                result = pattern[0].sub(pattern[1], result, re.VERBOSE)
            outfile.write(result)
    except (KeyboardInterrupt):
        pass
    except:
        raise
```

```
def main():
    parser = argparse.ArgumentParser(description=\
    """
        Prepare file[s]\n\n
        example: cat error_log | tail -n 1000 | ./%(prog)s
    """, formatter_class=RawDescriptionHelpFormatter)
    parser.add_argument('file', nargs='*', default=[],
                        help='file[s] to do some work')
    args = parser.parse_args()

    input_stream = fileinput.input(args.file)



    prepare(input_stream)

    pass

if __name__ == '__main__':
    main()
```

Приложение B

test_prepare.py

```
#!/usr/bin/env python
# encoding: utf-8

import unittest
import StringIO

from prepare import prepare


class TestSUBSTITUTION(unittest.TestCase):

    """Docstring for TestSUBSTITUTION. """

    def setUp(self):
        pass

    def test_UUID(self):
        text  = 'ACE088EB-ECA6-4348-905A-041EF10DBD53'
        text += 'AAAAAAAA-0000-FFFF-1111-999999999999'
        text += 'ACE088EB-ECA6-4348-905A-041EF10DBD53'
        text += 'ACE088EB-ECA6-4348-905A-041EF10DBDG3'

        result_text = StringIO.StringIO()
        prepare(text.split('\n'), result_text)

        reference_string  = 'UUID_SUBSTITUTE' * 3
        reference_string += 'ACE088EB-ECA6-4348-905A-041EF10DBDG3'

        self.assertEqual(result_text.getvalue(), reference_string)


if __name__ == '__main__':
    suite = unittest.TestLoader().loadTestsFromTestCase(TestSUBSTITUTION)
    unittest.TextTestRunner(verbosity=2).run(suite)
```

Приложение Г

mk_mixed.sh

```
#!/bin/bash

original_file=error_log
part_size=40000
mkdir -p tmp

echo "getting head of $original_file..."
head -n $part_size $original_file > tmp/$original_file.head

echo "getting tail of $original_file..."
tail -n $part_size $original_file > tmp/$original_file.tail

echo "merging head, tail and preparing with prepare.py..."
cat tmp/$original_file.head tmp/$original_file.tail |\
../prepare.py > $original_file.mixed.prep
echo "$original_file.mixed.prep ready"

echo "cleaning-up..."
rm -rf tmp/
```

Приложение Д

collect-logs.py

```
#!/usr/bin/env python2.7

import os
import errno
import getpass
import concurrent.futures
import subprocess
import shlex
import logging
import traceback

import argparse
import yaml

import time
from random import randint


COLLECT_LOG_DIR = "./logs/"


class GroupConfiguration(object):
    """
    Group configuration class. Contains remote hosts file paths and other usefull data.
    """

    DEFAULT_TMP = './logs/'

    def __init__(self, group, remote_tmp=None, local_tmp=None):
        """
        :param group: Group selector in format <group_name>:<port>. Port is used to generate ba
        :param remote_tmp: directory to store collected logs on remote hosts.
        :param local_tmp: directory to store collected logs on local side.
        """
```

```
        group_name, remote_log_path = self.parse_group(group)

        self.group_name = group_name
        self.remote_log_path = remote_log_path

        self.__local_tmp = local_tmp or self.DEFAULT_TMP
        self.__remote_tmp = remote_tmp or self.DEFAULT_TMP

    @property
    def remote_log_dir(self):
        return os.path.dirname(self.remote_log_path)

    @property
    def remote_log_file(self):
        return os.path.basename(self.remote_log_path)

    @property
    def local_log_dir(self):
        return self.__local_tmp

    @property
    def local_log_file(self):
        return self.remote_log_file

    @property
    def local_log_path(self):
        return os.path.join(self.local_log_dir, self.local_log_file)

    @property
    def remote_tmp_dir(self):
        return self.__remote_tmp

    @property
    def remote_tmp_file(self):
        return self.remote_log_file

    @property
    def remote_tmp_path(self):
        return os.path.join(self.remote_tmp_dir, self.remote_tmp_file)

    @staticmethod
```

```
    def parse_group(group):
        return group.strip().split(':')


class LogsCollector(object):

    def __init__(self, max_workers=3, aliases=None):
        self.executor = concurrent.futures.ThreadPoolExecutor(max_workers=max_workers)
        self.aliases = aliases or {}

    @staticmethod
    def run_command(cmd, comment=''):
        if isinstance(cmd, str):
            cmd = shlex.split(cmd)

        if comment:
            comment = '({0})'.format(comment)

        logging.debug("Running command '{}' {}...".format(cmd, comment))

        process = subprocess.Popen(cmd, shell=False)

        return process.wait()

    def remote_collect(self, group_config, size):
        """
        :param group_config: configuration of group: log paths, tmp file paths, etc.
        :type group_config: GroupConfiguration

        :param size: number of lines to collect from log on each host.
                     Result log size will be <size> * <hosts_number>.
        :type size: int

        :return: command
        :rtype: str
        """

        logging.info('Collecting logs on remote nodes...')

        remote_command = 'mkdir -p {tmp_dir};' \
                         'tail -n 0 -f {log_file} ' \
```

```
                '| head -n {size} > {tmp_file}'.format(tmp_dir=group_config.remote_tmp_dir
                                                     tmp_file=group_config.remote_tmp_path,
                                                     log_file=group_config.remote_log_path,
                                                     size=size)

    command = ['sky', 'run', '-U', remote_command, group_config.group_name]

    return self.run_command(command, 'remote_collect')

def local_collect(self, group_config):

    logging.info('Fetching logs to local side...')

    command = ['sky', 'download', '-U', group_config.remote_tmp_path,
               group_config.local_log_dir,
               group_config.group_name]

    return self.run_command(command, 'local_collect')

def remote_cleanup(self, group_config=None):

    logging.info('Cleaning remote caches...')

    remove_log = 'rm -f {}'.format(group_config.remote_tmp_path)
    # remove_dir = 'rmdir {} 2>/dev/null || true'.format(group_config.remote_tmp_dir)

    command = ['sky', 'run', '-U', remove_log, group_config.group_name]

    return self.run_command(command, 'remote_cleanup')

def collect_group(self, group_config, size):
    self.remote_collect(group_config=group_config, size=size)
    self.local_collect(group_config=group_config)
    self.remote_cleanup(group_config=group_config)

def collect(self, groups, size, remote_tmp=None, local_tmp=None):
    future_to_group = {}
    for group_spec in groups:
        if group_spec in self.aliases:
            future_to_group.update(
                {self.executor.submit(self.collect_group,
```

```
                                    GroupConfiguration(group, remote_tmp, local_tmp),
                                    size): group for group in self.aliases[group_spec]}
                )
            else:
                future = self.executor.submit(self.collect_group,
                                              GroupConfiguration(group_spec, remote_tmp, local_tmp),
                                              size)
                future_to_group[future] = group_spec

        for future in concurrent.futures.as_completed(future_to_group):
            group_spec = future_to_group[future]
            try:
                # import pdb; pdb.set_trace()
                future.result()
            except Exception as exc:
                logging.error("Group '{0}' treatment error: {1}".format(group_spec, exc))
                logging.error(traceback.format_exc())
            else:
                print("Group '{0}' treatment succeed.".format(group_spec))


def parse_args(args=None):
    parser = argparse.ArgumentParser(prog='collect-logs')

    parser.add_argument("-d", "--debug", action="store_const", dest="loglevel",
                        const="DEBUG", help="Set logging level to DEBUG")
    parser.add_argument("-v", "--verbose", action="store_const", dest="loglevel",
                        const="INFO", help="Set logging level to INFO")
    parser.add_argument("-q", "--quiet", action="store_const", dest="loglevel",
                        const="CRITICAL", help="Set logging level to CRITICAL")
    parser.add_argument("--loglevel", action="store", dest="loglevel",
                        type=str, help="Set logging level to LOGLEVEL")

    parser.add_argument('--threads', '-t', type=int, default=3)
    parser.add_argument('--size', '-s', type=int, default=1000)

    remote_tmp_default = "/home/{0}/logs".format(getpass.getuser())
    parser.add_argument('--remote-tmp', type=str, default=remote_tmp_default)
    parser.add_argument('--local-tmp', type=str, default='./logs/')

    parser.add_argument('groups', nargs='+', help='Hosts or groups for collect logs.')
```

```
    return parser.parse_args(args)


def convert_options(options):
    options.loglevel = options.loglevel.upper()


def read_config(config=None):
    if config is None:
        config = "./collect-logs.conf"

    if isinstance(config, str) and os.path.exists(config):
        config = file(config, 'r')

    return yaml.load(config)


def main(args=None):
    options = parse_args(args)

    convert_options(options)

    # Change logging level before any other actions.
    logging.root.setLevel(options.loglevel)

    config = read_config()

    collector = LogsCollector(options.threads, config['aliases'])
    collector.collect(options.groups, options.size, options.remote_tmp, options.local_tmp)


if __name__ == '__main__':
    main()
```