УДК 519.688, 004.42

Исследование многопоточности веб-сервера, разработанного на языке программирования Erlang

Кочетов Павел Сергеевич
Нижегородский государственный университет им. Н.И. Лобачевского
Магистрант

Штанюк Антон Александрович
Нижегородский государственный университет им. Н.И. Лобачевского
доцент

Аннотация
В статье рассматривается исследование многопоточности веб-серверов. Приводится реализация веб-сервера на языке программирования Erlang, анализируются результаты тестирования его многопоточности с помощью системы Tsung и сравниваются с результатами тестирования веб-сервера Apache.
Ключевые слова: язык программирования Erlang, сервер, исследование многопоточности веб-сервера, тестирование веб-сервера.

The study of multi-threading of a web server, developed in the programming language Erlang

Pavel Kochetov
Nizhny Novgorod State University, Nizhny Novgorod, Russia
Student

Anton Shtanyuk
Nizhny Novgorod State University, Nizhny Novgorod, Russia
associate professor

Abstract
The article discusses the study of multi-threading of web servers. Implementation of a web server in the programming language Erlang, analyzes results of testing of multi-threading using system Tsung and compared with the results of testing the Apache web server.
Keywords: Erlang programming language, server, the study of multi-threading web server, testing the web server.

Введение

В настоящее время веб-технологии используются повсеместно. Все больше становится веб-серверов, которыми ежедневно пользуются десятки

тысяч и даже миллионы пользователей одновременно, поэтому зачастую возникают серьезные проблемы в работе веб-серверов при их перегрузке. Однако технологии разработки программного обеспечения стремительно развиваются и сейчас существует множество вариантов решения данных проблем. Одним из вариантов решения является разработка веб-сервера на языке программирования Erlang с использованием многопоточности.

Erlang - это язык программирования, используемый для построения масштабируемых программных систем реального времени с учетом требований к высокой доступности [1].

Язык Erlang не так популярен среди разработчиков, как, например, Java, C#, Python. В настоящее время чаще всего используются объектно-ориентированные языки программирования, где все операции выполняются над объектами. В Erlang не используются объекты, их заменяют процессы. Возможно, из-за этих отличий и непривычного синтаксиса данный язык пока не приобрел известность в сообществе программистов.

Главное в Erlang - его уникальная легковесная модель параллельных вычислений. Она не зависит от операционной системы и способна справляться с огромным числом одновременно запущенных процессов. В Erlang виртуальная машина не создает отдельного потока вычислений операционной системы для каждого процесса, процессы создаются и выполняются внутри самой виртуальной машины и не зависят от операционной системы, в результате чего создание процесса занимает микросекунды и время создания не зависит от числа уже запущенных процессов, поэтому в Erlang процессы называют легковесными. За счет этого Erlang сильно превосходит такие языки, как Java и C#, в которых для каждого процесса выделяется отдельный поток вычислений операционной системы.

Erlang, в отличие от многих популярных языков программирования, избегает использования разделяемой памяти (shared memory). Благодаря этому, Erlang прекрасно работает на многоядерных процессорах, что позволяет решать проблемы с синхронизацией и возникновением узких мест. Программы, написанные на языке Erlang являются более краткими и элегантными в отличие от программ, написанных на объектно-ориентированных языках.

Erlang разрабатывался для создания распределенных, надежных, отказоустойчивых систем реального времени. Сейчас, язык используется многими крупными компаниями: Yandex, Facebook, Ericsson в своих проектах.

Архитектуру веб-сервера можно представить в виде следующих четырех уровней:

1. Принятие запросов и routing;
2. Бизнес-логика;
3. Хранение данных;
4. Длительные задачи отложенного выполнения (перекодировка видео, сбор и анализ логов).

На первом уровне Erlang отлично справляется со своей задачей, за счет своего механизма легковесных процессов. В ряде случаев таких процессов может быть создано сотни тысяч и даже миллионы.

На остальных трех уровнях Erlang также неплохо работает, однако, при разработке высоконагруженных систем обычно используют несколько разных технологий на разных уровнях. В этом случае Erlang можно использовать на первом уровне и в качестве соединителя компонентов системы.

Главное достоинство Erlang для веб-систем - простота (насколько это возможно) разработки распределенных систем.

Основные недостатки Erlang:

1. Плохая работа со строками;
2. Некачественная поддержка Unicode;
3. Динамическая типизация (частично компенсируется dialyzer);
4. Малое количество и незавершенность библиотек [2].

Также в [2] сравниваются реализации небольшого веб-проекта на Erlang и других языках программирования и рассматриваются преимущества Erlang-реализации проекта.

Для языка Erlang существует расширение Wooper, позволяющее применять ООП при разработке программ на Erlang так же, как в типичных ООП-языках программирования. Использование данного расширения может существенно упростить разработку сложной программной системы. В [3] приведена подробная документация по расширению Wooper для языка программирования Erlang.

Цель данной работы состоит в исследовании многопоточности веб-серверов.

Для достижения поставленной цели были поставлены и решены следующие задачи:

1. Разработка веб-сервера на языке программирования Erlang;
2. Установка и настройка системы нагрузочного тестирования Tsung;
3. Проведение тестов и анализ полученных результатов.

Обзор веб-серверов на Erlang и библиотек для их построения

В настоящее время на базе языка программирования Erlang существует несколько веб-серверов и фреймворков, упрощающих их создание. Рассмотрим основные из них подробнее.

Yaws

Yaws – это высокоэффективный HTTP-сервер, который очень хорошо подходит для веб-приложений с динамическим содержимым. Поддерживает два разных режима работы:

- Автономный режим, в котором Yaws работает как обычный веб-сервер, работающий в фоновом режиме. Это режим по умолчанию.
- Встроенный режим, в котором Yaws работает как встроенный веб-сервер в другом Erlang-приложении.

Yaws полностью написан на Erlang, и, кроме того, является многопоточным веб-сервером, где для обработки каждого клиента используется лишь по одному легковесному процессу Erlang.

Основными преимуществами Yaws по сравнению с другими веб-серверами являются производительность и элегантность. Производительность исходит из базовой системы Erlang и ее возможности эффективной параллельной обработки нескольких процессов. Элегантность веб-сервера Yaws также исходит из особенностей языка программирования Erlang [4].

Cowboy

Cowboy - это компактный, быстрый и современный HTTP-сервер для Erlang/OTP.

Cowboy предоставляет полный стек возможностей HTTP в своем небольшом исходном коде. Он оптимизирован для низкой латентности и низкого уровня использования памяти, поскольку он использует двоичные строки. Двоичные строки более эффективны для представления строк, чем списки, так как они занимают меньше места в памяти. Производительность обработки может варьироваться в зависимости от операции. Как известно, двоичные строки используются для получения ускорения, если код изначально скомпилирован.

Еще одна особенность Cowboy – это способ отправки даты и времени в HTTP-заголовках. Cowboy определяет текущую дату и время каждую секунду и разделяет это значение на все одновременные процессы, что позволяет улучшить производительность.

По умолчанию установлено очень большое максимальное количество активных соединений. Это необходимо для того, чтобы предотвратить слишком большое количество процессов, выполняющих потенциально тяжелую работу, замедляя все остальные процессы или занимая всю память.

Cowboy предоставляет возможности маршрутизации, позволяет выборочно посылать запросы на обработчики, написанные на Erlang. Веб-сервер Cowboy может быть легко интегрирован в любое другое приложение [5].

MochiWeb

MochiWeb – это библиотека, написанная на Erlang для построения легковесных HTTP-серверов. Несмотря на отсутствие официального сайта и подробной официальной документации, MochiWeb является идеальной

отправной точкой для построения веб-сервисов на языке Erlang, способных выдерживать нагрузку до сотен тысяч одновременных параллельных соединений.

Misultin

Misultin – еще одна библиотека для построения легковесных веб-серверов. Misultin может быть использован как для создания автономного веб-сервера, так и для интеграции в другие приложения. Однако проект был заморожен автором, так как Misultin, Cowboy и MochiWeb реализуют очень схожий функционал.

Inets

Inets – приложение, входящее в состав Erlang/OTP, в состав которого в виде служб входят: HTTP-сервер, HTTP-клиент, TFTP-сервер, TFTP-клиент и FTP-клиент.

HTTP-сервер на базе приложения Inets имеет в своем составе следующий функционал: обработка запросов, логирование, аутентификация, запуск CGI-скриптов и фильтрация запросов.

Обзор системы нагрузочного тестирования Tsung

Для тестирования многопоточности веб-серверов была использована тестирующая система Tsung. Рассмотрим подробнее ее основные возможности.

Tsung — это распределенная система нагрузочного тестирования, разработанная на Erlang. Tsung поддерживает работу с HTTP, WebDAV, SOAP, PostgreSQL, MySQL, LDAP и XMPP/Jabber.

Главные возможности Tsung:

- Высокая производительность: Tsung способен симулировать огромное количество одновременных пользователей с одного компьютера;
- Распределенность: нагрузку можно распределить на группу клиентских машин;
- Возможность использования разных протоколов, используя систему плагинов: HTTP, WebDAV, SOAP, XMPP/Jabber и другие;
- Поддержка SSL;
- Возможность использования нескольких ip-адресов на одном компьютере;
- Мониторинг операционной системы (процессора, оперативной памяти и сетевого траффика), используя Erlang-агентов на удаленных серверах или SNMP;

- Система конфигурации на XML: сценарии пользователей записываются в XML-документ. Для записи сценариев через браузер можно использовать Tsung Recorder (только для HTTP и PostgreSQL);
- Динамические сценарии: можно получить динамические данные с сервера под нагрузкой (без написания кода) и использовать их в последующих запросах. Также присутствует возможность использования циклических запросов, остановки или перезапуска сессии в зависимости от полученного ответа с сервера;
- Смешанное поведение: несколько сессий могут быть использованы для имитации различных типов пользователей в течение одного теста;
- Стохастические процессы: для генерации реалистичной нагрузки пользовательские параметры (thinktime, arrival rate) могут быть заданы случайно с помощью распределения вероятностей (экспоненциального).

Возможности системы для протокола HTTP:

- Поддержка HTTP/1.0 и HTTP/1.1;
- Запросы: GET, POST, PUT, DELETE, HEAD, OPTIONS, PATCH;
- Куки: автоматическое или ручное управление куками;
- Прохождение аутентификации;
- Возможность добавления HTTP-заголовков;
- Возможность записи сессии пользователя с помощью браузера;
- Поддержка SOAP в режиме HTTP
- Возможность нагрузочного тестирования HTTP-сервера [6].

В Tsung есть возможность построения отчетов и графиков по записанным в ходе тестирования логам после завершения нагрузочного теста или во время его проведения.

Генерируемые системой отчеты и графики содержат следующие статистические данные:

- Производительность: время отклика, время соединения, декомпозиция сценария пользователя на основе группировки запросов, число запросов в секунду;
- Ошибки: статистика кодов возврата страниц и коды ошибок.

Настройка необходимого окружения для тестирования

Для тестирования многопоточности веб-серверов на подготовительном этапе было выполнено:

- Создание виртуальной машины в программе VMware на базе дистрибутива CentOS 7 x64;
- Установка Erlang/OTP R16B02;
- Установка тестирующей системы Tsung вместе с необходимыми для нее зависимостями;
- Настройка конфигурации приложения Tsung.

На этапе создания и настройки виртуальной машины CentOS 7 для нее было выделено следующее количество ресурсов:

- 5.1 GB RAM;
- 4-х ядерный процессор AMD A8-6410;
- 20 GB на HDD.

Настройка конфигурации приложения Tsung

Система нагрузочного тестирования Tsung настраивается с помощью одного XML-документа, который называется `tsung.xml`.

Пример структуры XML-файла конфигурации приложения Tsung выглядит следующим образом:

```
<?xml version="1.0"?>
<tsung loglevel="info" dumptraffic="false" version="1.0">
  <clients>
    <client    host="localhost"    use_controller_vm="true"
maxusers="30000000"/>
  </clients>
  <servers>
    <server host="localhost" port="80" type="tcp"/>
  </servers>
  <load duration="4" unit="minute">
   <arrivalphase phase="1" duration="2" unit="minute">
     <users arrivalrate="5000" unit="second"/>
   </arrivalphase>
   <arrivalphase phase="2" duration="2" unit="minute">
     <users arrivalrate="10000" unit="second"/>
   </arrivalphase>
  </load>

 <sessions>
  <session name="bot" probability="100" type="ts_http">
    <request>    <http    url="/index.html"    method="GET"
version="1.1"/> </request>
    <request>    <http    url="/index.html"    method="GET"
version="1.1"/> </request>
    <request>    <http    url="/index.html"    method="GET"
version="1.1"/> </request>
    <request>    <http    url="/index.html"    method="GET"
version="1.1"/> </request>
    <request>    <http    url="/index.html"    method="GET"
version="1.1"/> </request>
  </session>
 </sessions>
</tsung>
```

Главный тег в конфигурационном файле – `<tsung>`, внутри которого располагаются все остальные необходимые теги с настройками. У данного

тега существует 3 атрибута: `version` (версия ядра системы), `loglevel` (уровень подробности записи логов в ходе тестирования) и `dumptraffic` (используется для отладки).

Тег `<clients>` необходим для настройки эмулируемого клиента. Клиентов может несколько; каждый из них указывается с помощью вложенного тега `<client>`. У каждого тега `<client>` в данном выше примере указано 3 атрибута: `host` (имя хоста, по умолчанию `localhost`), `use_controller_vm` (использование контроллера виртуальной машины Erlang) и `maxusers` (максимальное количество пользователей, имитируемых системой, по умолчанию равно 800).

Внутри тега `<servers>` с помощью вложенных тегов `<server>` указываются тестируемые сервера. В данном случае задан 1 тестируемый сервер с параметрами: `host="localhost"` (имя хоста), `port="80"` (порт) и `type="tcp"` (используемый сетевой протокол).

Необходимая нагрузка устанавливается с помощью тега `<load>`. В приведенном выше примере тег имеет 2 параметра: `duration="4"` и `unit="minute"`. Данная конфигурация означает, что нагрузка на указанный ранее сервер будет осуществляться в течении 4 минут. Нагрузку можно разбивать на несколько фаз с помощью тегов `<arrivalphase>`, у каждого из которых должны быть указаны следующие параметры: `phase` (номер фазы), `duration` (длительность во временных единицах, указанных в атрибуте `unit`) и `unit` (временная единица, например: `hour` – час, `minute` – минута, `second` – секунда). Внутри каждого тега `<arrivalphase>` указан тег `<users>`, который отвечает за количество генерируемых системой пользователей в единицу времени. Таким образом, строка `<users arrivalrate="5000" unit="second"/>` будет означать, что каждую секунду будет приходить 5000 новых пользователей.

В теге `<sessions>` указываются все возможные сессии клиентов (последовательности действий клиентов на тестируемом сервере), приходящих на сервер. В нашем примере внутри тега `<sessions>` находится только 1 тег `<session>` с параметрами: `name="bot"` (имя сессии), `probability="100"` (вероятность выполнения приходящим клиентом данной сессии в процентах; при указании нескольких разных сессий в зависимости от значения этого атрибута для каждого приходящего на сервер клиента будет выбираться та или иная последовательность действий) и `type` (для тестирования HTTP-сервера должно быть указано значение `ts_http`).

В каждой заданной сессии указывается последовательность запросов, которые выполняют приходящие на сервер пользователи. Например, строка `<request> <http url="/index.html" method="GET" version="1.1"/> </request>` означает, что будет выполнен GET-запрос на получение статической страницы по пути `localhost/index.html`.

Описание реализации HTTP-сервера на Erlang

Для реализации простого HTTP-сервера на базе приложения Inets, входящего в состав Erlang/OTP был создан файл `http_server_on_inets.erl`.

В начале файла обязательно должно быть указано название реализуемого модуля, причем оно должно совпадать с именем файла:

```
-module(http_server_on_inets).
```

Далее указываются все методы, реализованные в данном модуле с указанием числа их входных параметров для доступа к ним извне:

```
-export([start/0]).
```

Затем описываются реализуемые методы. В нашем случае единственный метод `start()`:

```
start() ->
  inets:start(),
  Pid = inets:start(httpd, [
    {modules, [
      mod_alias,
      mod_auth,
      mod_esi,
      mod_actions,
      mod_cgi,
      mod_dir,
      mod_get,
      mod_head,
      mod_log,
      mod_disk_log
    ]},
    {port, 8082},
    {server_name, "http_server_on_inets"},
    {server_root,
"/home/pav/Документы/http_server_on_inets"},
    {document_root,
"/home/pav/Документы/http_server_on_inets/www"},
    {directory_index, ["index.html"]},
    {max_clients, 3000000},
    {erl_script_alias, {"/erl", [http_server_on_inets]}},
    {error_log, "error.log"},
    {security_log, "security.log"},
    {transfer_log, "transfer.log"},
    {mime_types, [
      {"html", "text/html"},
      {"css", "text/css"},
      {"js", "application/x-javascript"}
    ]}
  ]),
  io:fwrite("~p\r\n", [Pid]).
```

Рассмотрим приведенный код подробнее.

`start() ->` - начало реализации метода.

`inets:start()` - запуск приложения Inets.

`Pid = inets:start(httpd, […])` - запуск HTTP-сервера на основе приложения Inets, `Pid` – идентификатор запускаемого процесса, в котором стартует сервер.

`{modules, […]}` - список модулей, используемых внутри запускаемого HTTP-сервера.

`{port, 8082}` - для запуска сервера будет использоваться порт 8082.

`{server_name, "http_server_on_inets"}` - имя сервера.

`{server_root, "/home/pav/Документы/http_server_on_inets"}` - корневая директория сервера.

`{document_root, "/home/pav/Документы/http_server_on_inets/www"}` - корневая директория расположения HTML-документов.

`{directory_index, ["index.html"]}` - HTML-страница, которая будет открываться по умолчанию при доступе к серверу.

`{max_clients, 3000000}` - максимальное число одновременных клиентов данного сервера.

`{erl_script_alias, {"/erl", [http_server_on_inets]}}` - алиас для использования Erlang-кода на сервере.

`{error_log, "error.log"}` – лог ошибок (путь к файлу относительно корневой директории сервера).

`{security_log, "security.log"}` – лог модуля фильтрации `mod_security` (путь к файлу относительно корневой директории сервера).

`{transfer_log, "transfer.log"}` – лог всех запросов на сервере (путь к файлу относительно корневой директории сервера).

`{mime_types, […]}` – типы документов, которые может обрабатывать веб-сервер.

`io:fwrite("~p\r\n", [Pid])` – вывод идентификатора процесса, в котором запущен сервер на консоль.

Тестирование веб-серверов и анализ полученных результатов

В ходе тестирования многопоточности веб-серверов была проведена серия экспериментов для разработанного веб-сервера и веб-сервера Apache.

Для тестирования разработанного веб-сервера в системе Tsung была задана следующая конфигурация:

```
<load duration="4" unit="minute">
  <arrivalphase phase="1" duration="2" unit="minute">
    <users arrivalrate="500" unit="second"/>
  </arrivalphase>
  <arrivalphase phase="2" duration="2" unit="minute">
    <users arrivalrate="1000" unit="second"/>
  </arrivalphase>
</load>
```

```
      <sessions>
       <session name="bot" probability="100" type="ts_http">
         <request>    <http    url="/index.html"    method="GET"
version="1.1"/> </request>
       </session>
      </sessions>
```

Таким образом, общее время выполнения теста – 4 минуты. Тест состоит из 2 фаз.

Первая фаза: продолжительность – 2 минуты, каждую секунду на сервер приходит 500 новых пользователей.

Вторая фаза: продолжительность – 2 минуты, каждую секунду на сервер приходит 1000 новых пользователей.

Каждый пришедший пользователь отправляет GET-запрос на получение страницы с `url="localhost:8082/index.html"`.

После выполнения теста с заданными параметрами конфигурации были получены следующие результаты:

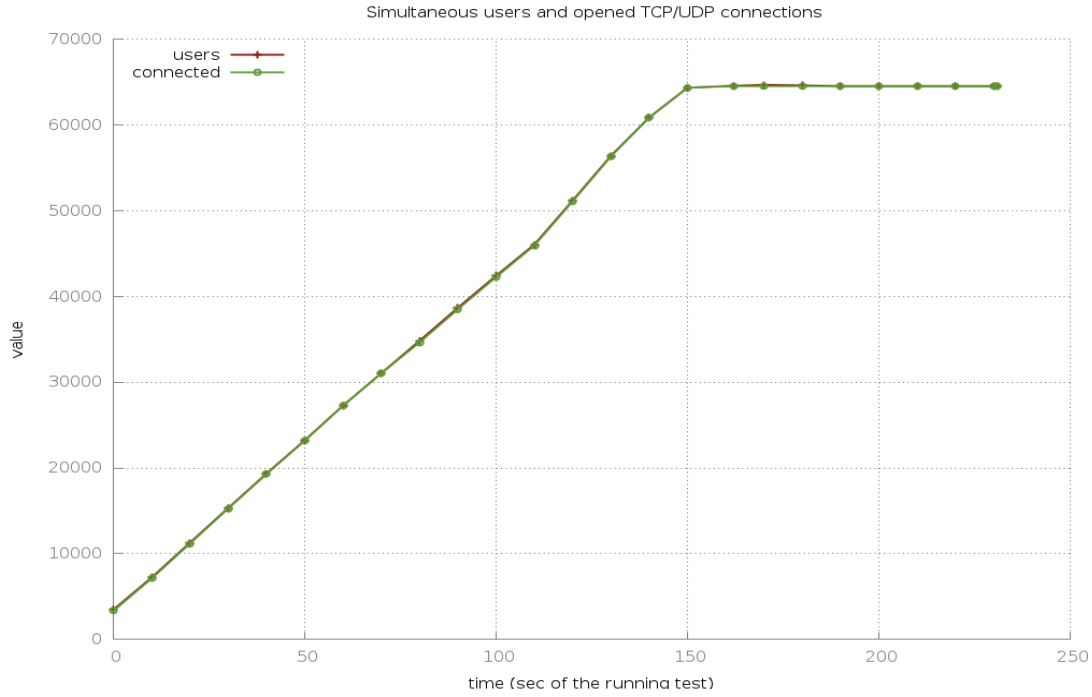

Рисунок 1 - Число одновременных пользователей и открытых TCP/UDP соединений

На рисунке 1 можно видеть, что на протяжении всего времени проведения теста число подключенных одновременно пользователей постоянно возрастает, однако, начиная со 150-ой секунды теста возрастание числа одновременных пользователей прекращается.

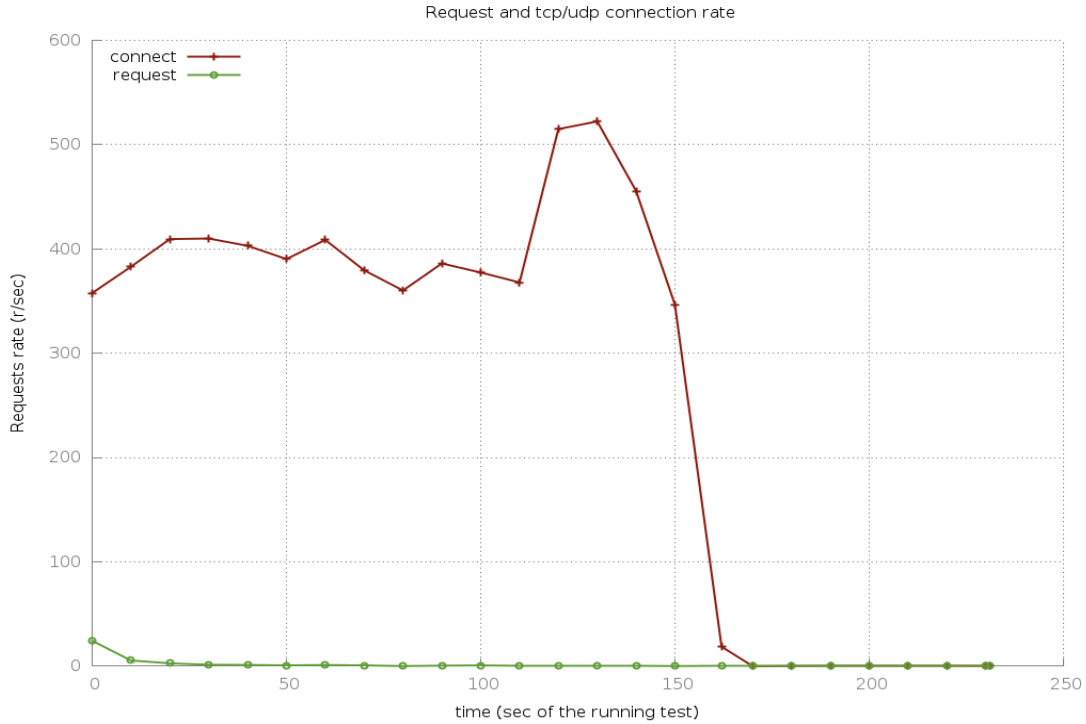

Рисунок 2 - Число обрабатываемых запросов и TCP/UDP подключений в секунду

На рисунке 2 можно заметить, что число обрабатываемых запросов в секунду быстро снижается. Число TCP/UDP подключений в секунду начинает резко снижаться после 150-й секунды выполнения теста.

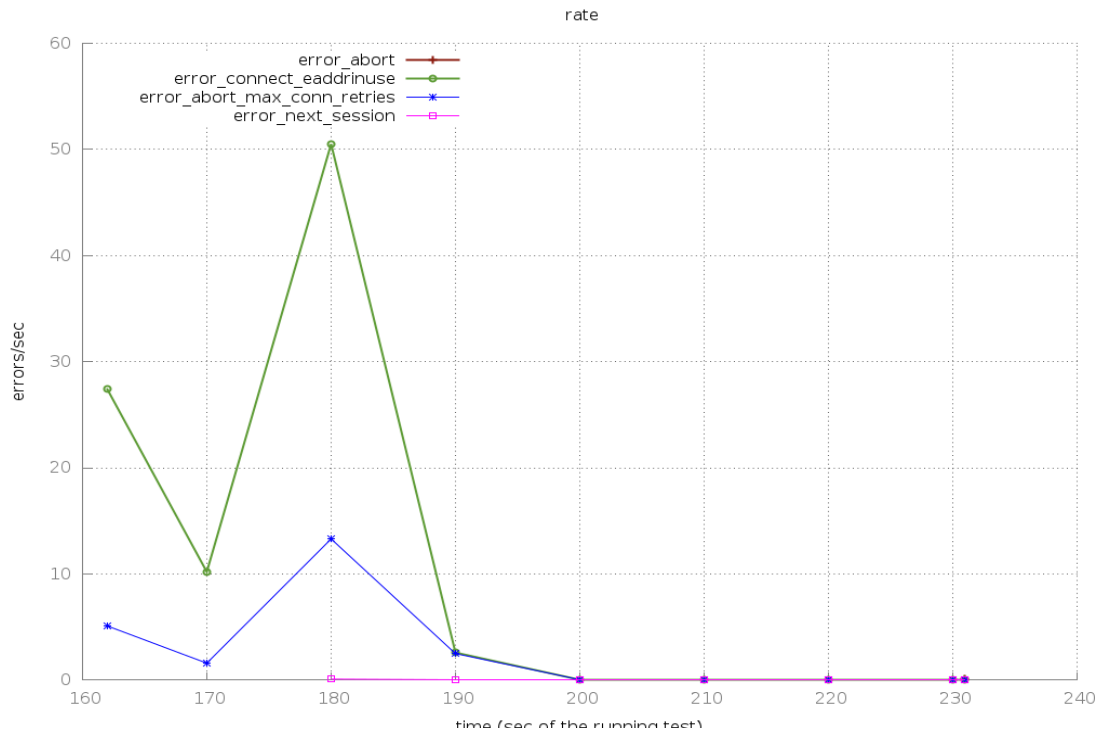

Рисунок 3 - Число ошибок в секунду

На рисунке 3 видно, что после 160-й секунды выполнения теста начинают появляться ошибки в работе сервера, что говорит о том, что с этого момента времени сервер перестает справляться с нагрузкой.

По результатам проведенного теста многопоточности простого сервера, разработанного на Erlang, можно сделать вывод, что реализованный сервер способен выдерживать нагрузку около 65000 одновременных подключений клиентов к нему.

Далее было проведено тестирование многопоточности сервера Apache. Для тестирования веб-сервера Apache использовалась следующая конфигурация системы Tsung:

```
<load duration="4" unit="minute">
   <arrivalphase phase="1" duration="2" unit="minute">
     <users arrivalrate="20000" unit="second"/>
   </arrivalphase>
   <arrivalphase phase="2" duration="2" unit="minute">
     <users arrivalrate="40000" unit="second"/>
   </arrivalphase>
 </load>

 <sessions>
  <session name="bot" probability="100" type="ts_http">
    <request>    <http    url="/index.html"    method="GET"
version="1.1"/> </request>
  </session>
 </sessions>
```

Таким образом, общее время выполнения теста – 4 минуты. Тест состоит из 2 фаз.

Первая фаза: продолжительность – 2 минуты, каждую секунду на сервер приходит 20000 новых пользователей.

Вторая фаза: продолжительность – 2 минуты, каждую секунду на сервер приходит 40000 новых пользователей.

Каждый пришедший пользователь отправляет GET-запрос на получение страницы с `url="localhost/index.html"`.

После выполнения теста с заданными параметрами конфигурации были получены следующие результаты.

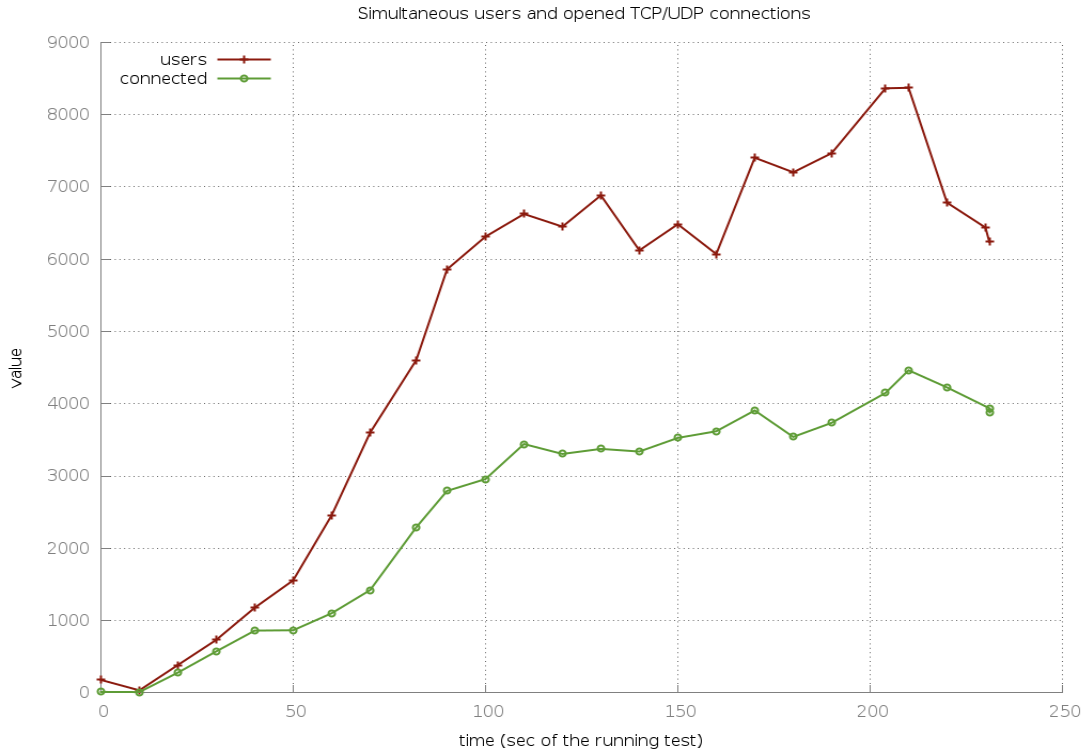

Рисунок 4 - Число одновременных пользователей и открытых TCP/UDP соединений (Apache)

На рисунке 4 можно видеть, что на протяжении всего времени проведения теста число подключенных одновременно пользователей постоянно возрастает, однако, за все время проведения теста количество одновременно подключенных пользователей не превышает 4500.

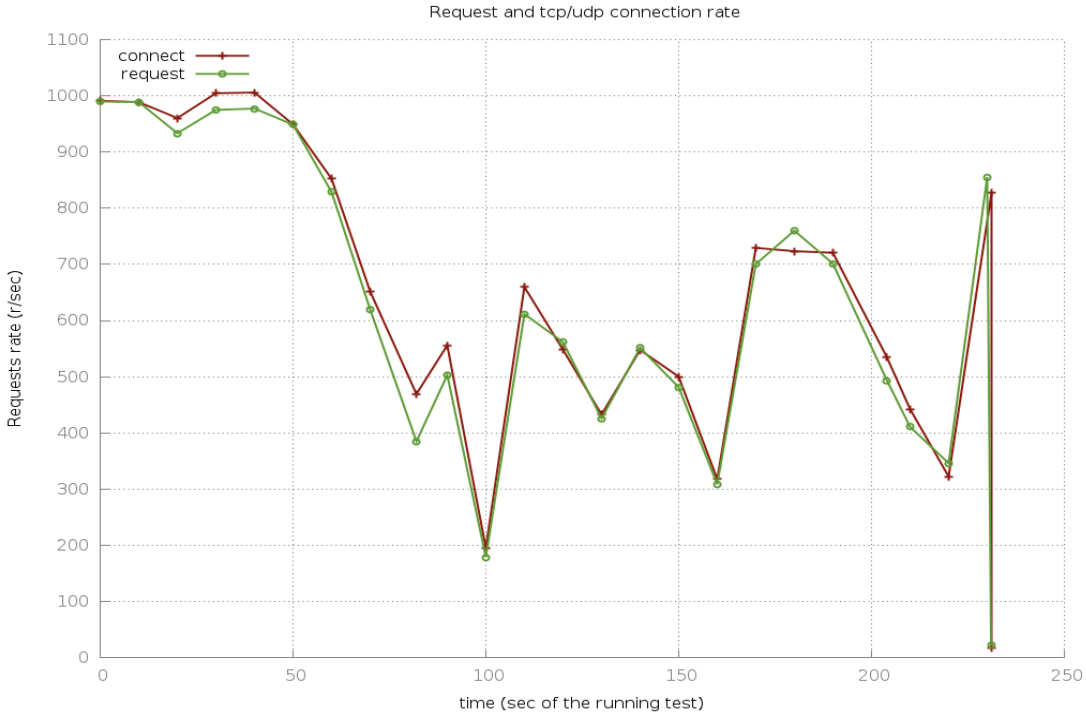

Рисунок 5 - Число обрабатываемых запросов и TCP/UDP подключений в секунду (Apache)

На рисунке 5 можно заметить, что число обрабатываемых запросов в секунду после первых 50 секунд выполнения теста снижается и затем начинает колебаться в диапазоне 200-800 запросов в секунду. График числа TCP/UDP подключений в секунду ведет себя аналогичным образом.

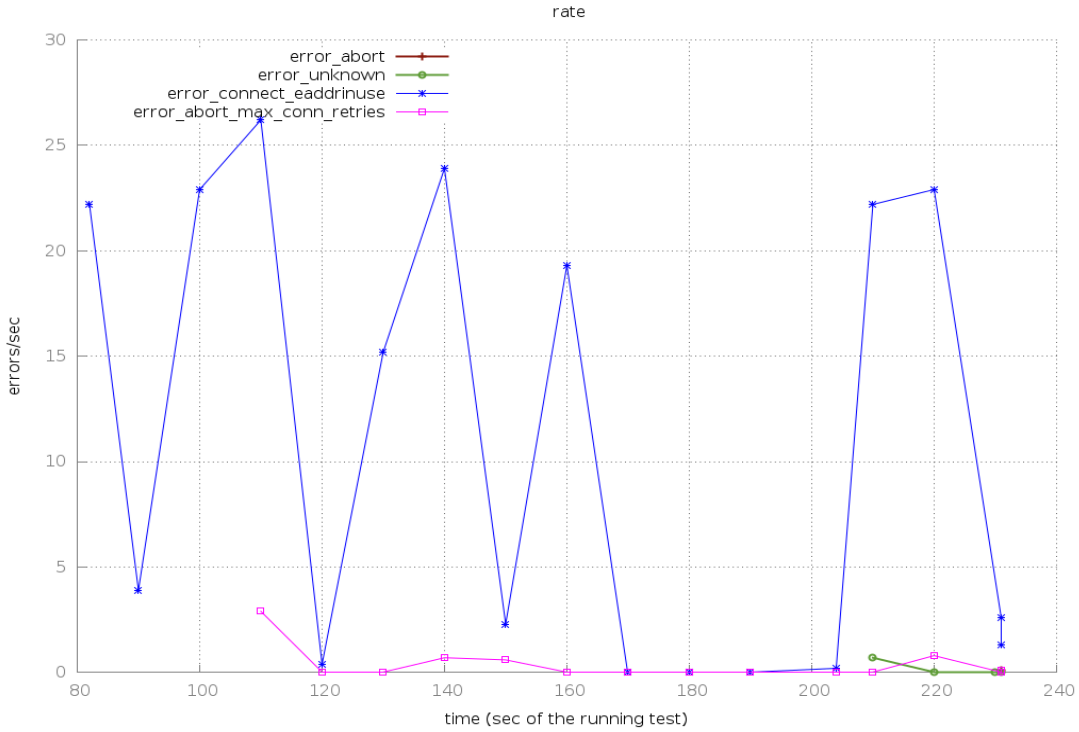

Рисунок 6 - Число ошибок в секунду (Apache)

На рисунке 6 видно, что после 80-й секунды выполнения теста начинают появляться ошибки в работе сервера, что говорит о том, что с этого момента времени сервер перестает справляться с нагрузкой.

По результатам проведенного теста многопоточности веб-сервера Apache можно сделать вывод, что он способен выдерживать нагрузку около 3000-4000 одновременных подключений клиентов.

Таким образом, разработанный на Erlang веб-сервер способен поддерживать одновременных подключений примерно в 15 раз больше, чем веб-сервер Apache.

Выводы

В ходе данной работы было проведено исследование многопоточности веб-серверов. Для этого был разработан простой веб-сервер на Erlang, установлено и настроено приложение для нагрузочного тестирования Tsung и проведены соответствующие эксперименты с разработанным веб-сервером и сервером Apache. Выяснилось, что разработанный сервер способен выдерживать примерно в 10 раз больше одновременных подключений клиентов, чем сервер Apache, что говорит о высокой масштабируемости сервера, разработанного на Erlang.

Библиографический список

1. Erlang Programming Language. URL: http://www.erlang.org/
2. Кочетов, П.С. Использование языка программирования Erlang при разработке web-серверов / П.С. Кочетов, А.А. Штанюк // Постулат. 2016. №12. URL: http://e-postulat.ru/index.php/Postulat/article/view/263/278.
3. WOOPER. Wrapper for Object-Oriented Programming in Erlang. URL: http://ceylan.sourceforge.net/main/documentation/wooper/index.html.
4. Yaws. URL: http://yaws.hyber.org/
5. Cowboy 2.0 User Guide. URL: https://ninenines.eu/docs/en/cowboy/2.0/guide/introduction/
6. Tsung Documentation. URL: http://tsung.erlang-projects.org/user_manual.pdf.