УДК 519.688, 004.42

Использование языка программирования Erlang при разработке веб-серверов

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 рассматриваются возможности языка программирования Erlang при разработке производительных web-серверов. Особое внимание уделяется поддержке объектно-ориентированного подхода. Приводится пример разработки класса на основе расширения Wooper.
Ключевые слова: язык программирования Erlang, сервер, объектно-ориентированный подход.

Web Servers Development with Erlang

Pavel Kochetov
Nizhny Novgorod State University
student

Anton Shtanyuk
Nizhny Novgorod State University
associate professor

Abstract
The article discusses the possibility of Erlang programming language in the development of the productive web-servers. Particular attention is paid to support object-oriented approach. An example of a class of development using Wooper extention.
Keywords: Erlang programming language, server, object-oriented approach.

Введение

Цель данной работы состоит в анализе средств языка Erlang для возможности использования объектно-ориентированного подхода при разработке многопоточного отказоустойчивого веб-сервера.

Особенности Erlang

Erlang - функциональный язык программирования с сильной динамической типизацией, предназначенный для создания распределенных вычислительных систем [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 [2]. 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 [2].

Erlang для веб-разработки

Часто при проектировании современных веб-систем необходимо учитывать возможные высокие нагрузки на системы. Например: десятки тысяч пользователей потребуют десятки тысяч сетевых соединений, сотни тысяч запросов к базе данных - все это надо обслуживать одновременно.

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 web-систем - простота (насколько это возможно) разработки распределенных систем.

Основные недостатки Erlang:

1. Плохая работа со строками;
2. Некачественная поддержка Unicode;
3. Динамическая типизация, которая частично компенсируется использованием dialyzer;
4. Малое количество и незавершенность библиотек.

Рассмотрим пример использования Erlang в небольшом web-проекте. Campfire - групповой web-чат для команды разработчиков. Изначально проект был разработан на RubyOnRails/MySQL, однако серверная часть не справлялась с нагрузкой в 1500 запросов в секунду. Позднее была разработана новая версия проекта на C. Эта версия справлялось с требуемой нагрузкой, но не была масштабируема. Затем была разработана версия программы на Erlang, которая справлялась с нагрузками не хуже версии, реализованной на C и при этом была масштабируема.

В [3] приведена сравнительная таблица параметров системы, написанной на различных языках программирования.

Таблица 1. Параметры серверов при использовании различных языков

	Ruby	C	Erlang
LOC	127	397	273
Req/sec	250-350	1800	1800
Responce	20ms	2-3ms	2-3ms
OS Processes	n/a	80	1
Extensible	Yes	No	Yes

Таким образом, в некоторых задачах Erlang демонстрирует производительность, сопоставимую с C. При этом код программы на Erlang более понятный, короткий и поддерживаемый. Однако стоит отметить, что Erlang использует по 1 процессу на 1 ядро CPU.

Поддержка объектно-ориентированного подхода (ООП) в Erlang

Для языка Erlang существует расширение Wooper, позволяющее применять ООП при разработке программ на Erlang так же, как в типичных ООП-языках программирования [5]. Рассмотрим основные возможности данного расширения на примере класса class_Cat:

1) Создание класса:

```
-module(class_Cat).
```

2) Создание модуля с названием класса и определение родителей данного класса (если они имеются):

```
-define( wooper_superclasses,
[class_Mammal,class_ViviparousBeing] ).
```

3) Определение параметров конструктора создаваемого класса:

```
-define( wooper_construct_parameters, Age, Gender,
FurColor, WhiskerColor ).
```

4) Определение стандартных возможностей расширения Wooper:

```
-define( wooper_construct_export, new/4, new_link/4,
 synchronous_new/4, synchronous_new_link/4,
 synchronous_timed_new/4, synchronous_timed_new_link/4,
 remote_new/5, remote_new_link/5,
remote_synchronous_new/5,
 remote_synchronous_new_link/5,
remote_synchronous_timed_new/5,
 remote_synchronous_timed_new_link/5, construct/5,
delete/1 ).
```

5) Определение методов, которые будут реализованы в данном классе:

```
-define( wooper_method_export, getTeatCount/1,
canEat/2, getWhiskerColor/1 ).
```

6) Подключение расширения WOOPER с помощью библиотеки wooper.hrl:

```
-include("wooper.hrl").
```

7) Описание конструктора создаваемого класса: вызов конструкторов родительских классов, указание специфических атрибутов:

```
construct( State, ?wooper_construct_parameters ) ->
 MammalState = class_Mammal:construct( State, Age,
Gender, FurColor ),
 ViviparousMammalState =
class_ViviparousBeing:construct( MammalState ),

 ?setAttribute( ViviparousMammalState, whisker_color,
WhiskerColor ).
```

8) Реализация методов нового класса:

```
delete(State ) ->
 io:format( "Deleting cat ~w! (overridden
destructor)~n", [self()] ),
 State.

getTeatCount(State) ->
 ?wooper_return_state_result( State, 6 ).

canEat(State,soup) ->
 ?wooper_return_state_result( State, true );

canEat(State,chocolate) ->
 ?wooper_return_state_result( State, true );

canEat(State,croquette) ->
 ?wooper_return_state_result( State, true );

canEat(State,meat) ->
 ?wooper_return_state_result( State, true );

canEat(State,_) ->
 ?wooper_return_state_result( State, false ).

getWhiskerColor(State)->
 ?wooper_return_state_result( State,
?getAttr(whisker_color) ).
```

В данном примере рассмотрено создание класса и множественное наследование. Однако расширение Wooper предусматривает и полиморфизм. Так, например, если в классе class_Cat не был бы определен метод getAge, то все равно для экземпляра данного класса можно было бы использовать данный метод, так как он определен в одном из классов-родителей (в данном случае class_Creature). Соответственно, в зависимости от класса экземпляра,

всегда будет вызван нужный метод соответствующего класса. Чтобы обеспечить инкапсуляцию, класс, создаваемый с помощью расширения Wooper сопоставляется модулю Erlang, имя которого по соглашению начинается с префикса "class_", за которым следует имя класса без разделителей (например, class_Creature). Благодаря тому, что для каждого экземпляра есть ассоциативная таблица, где ключами являются имена атрибутов, а значениями являются значения атрибутов, также обеспечивается инкапсуляция.

Выводы

Рассмотрев особенности языка программирования Erlang, выяснилось, что Erlang отлично подходит для разработки многопоточного отказоустойчивого веб-сервера, так как язык обладает уникальной легковесной моделью параллельных вычислений и превосходной отказоустойчивостью по сравнению с другими технологиями разработки. Также в Erlang для удобства можно использовать объектно-ориентированный подход при написании программ с помощью расширения Wooper, что может существенно упростить реализацию, за счет структурирования и снижения повторного использования кода.

Библиографический список

1. Википедия [Электронный ресурс.] - Режим доступа: https://ru.wikipedia.org/wiki/Erlang (дата обращения 29.11.2016).
2. Томпсон С., Чезарини Ф. Программирование в Erlang/Пер. с англ. Холомьева А. О. - М.: ДМК Пресс, 2015.
3. CampfireLovesErlang [Электронный ресурс.] - Режим доступа: http://www.erlang-factory.com/upload/presentations/119/MarkImbriaco-ErlangFactoryLondon2009-CampfireLovesErlang.pdf (дата обращения 29.11.2016).
4. Опыт использования Erlang в разработке многопользовательской игры [Электронный ресурс.] - Режим доступа: http://erlang-russian.org/post/148 (дата обращения: 29.11.2016)
5. WOOPER. Wrapper for Object-Oriented Programming in Erlang [Электронный ресурс.] - Режим доступа: http://ceylan.sourceforge.net/main/documentation/wooper/index.html (дата обращения 11.12.2016).
6. Хеберт Ф. Изучай Erlang во имя добра. М.: ДМК Пресс, 2015.