Erlang

Erlang — функциональный язык программирования с сильной динамической типизацией, предназначенный для создания распределённых вычислительных систем. Разработан и поддерживается компанией Ericsson. Язык включает в себя средства порождения параллельных легковесных процессов и их взаимодействия через обмен асинхронными сообщениями в соответствии с моделью акторов.

Erlang был целенаправленно разработан для применения в распределённых, отказоустойчивых, параллельных системах реального времени, для которых, кроме средств самого языка, имеется стандартная библиотека модулей и библиотека шаблонных решений (так называемых поведений) — фреймворк OTP (англ. Open Telecom Platform). Программа на Erlang транслируется в байт-код, исполняемый виртуальными машинами, находящимися на различных узлах распределённой вычислительной сети. Erlang-системы поддерживают горячую замену кода, что позволяет эксплуатировать оборудование безостановочно.

Свой синтаксис и некоторые концепции Erlang унаследовал от языка логического программирования Пролог. Язык поддерживает многие типы данных, условные конструкции, сопоставление с образцом, обработку исключений, списковые включения и выражения битовых строк, функции (анонимные функции, функции высшего порядка, рекурсивные определения функций, оптимизацию хвостовой рекурсии), модули, приём и отправку сообщений между процессами. Препроцессор поддерживает работу с макросами и включение заголовочных файлов.

Популярность Erlang начала расти в связи с расширением его области применения (телекоммуникационные системы) на высоконагруженные параллельные распределённые системы, обслуживающие миллионы пользователей WWW, такие как чаты, системы управления содержимым, веб-серверы и распределённые, требующие масштабирования базы данных. Erlang применяется в нескольких NoSQL-базах данных высокой доступности.

История

В середине 1980-х в компьютерной лаборатории компании Ericsson исследовали применимость существовавших на тот момент языков программирования для программного обеспечения телекоммуникационных систем. Джо Армстронг (Joe Armstrong), Роберт Вирдинг (Robert Virding) и Майк Вильямс (Mike Williams) под руководством Бьярне Деккера (Bjarne Däcker), написав прототипы программ на различных языках, пришли к выводу, что ни один из этих языков не имел полного набора возможностей, необходимых в области телекоммуникационных систем. В результате был создан новый язык программирования — Erlang. Своё название язык, вероятно, получил в честь датского математика и инженера Агнера Эрланга, основателя научного направления по изучению сетевого трафика в телекоммуникационных системах. По другой версии, название языка изначально было сокращением от «ericsson language».

Влияние на Erlang оказали ML, Миранда, Ада, Модула-2, CHILL, Пролог. Кроме того, на способ обновления программного обеспечения повлиял Smalltalk и использованные Ericsson проприетарные языки EriPascal и PLEX.

Потребовалось четыре года развития языка и прототипирования с использованием виртуальной машины Пролога, после чего в 1991 году Майк Вильямс переписал виртуальную машину для Erlang на Си. В 1992 году Erlang был впервые использован в коммерческом проекте. В 1995 году вышел новый релиз Erlang, вобравший накопившийся к тому моменту опыт

использования языка. Язык был сочтён достаточно развитым для использования в других продуктах компании (решения для широкополосной связи, GPRS, ATM).

В декабре 1995 года случилось событие, которое Джо Армстронг считает решающим для Erlang: проект AXE-N в Ellemtel по созданию нового маршрутизатора (как оборудования, так и системного программного обеспечения на C++) потерпел неудачу. В результате реорганизации проекта удалось, использовав разработанное оборудование и язык программирования Erlang, начать работы над ATM-маршрутизаторами серии AXD. Ресурсов лаборатории для такого проекта оказалось недостаточно, поэтому для работ по Erlang было создано производственное подразделение под названием OTP (Open Telecom Platform). В 1996 году увидел свет одноимённый фреймворк OTP.

Неожиданно, в 1998 году топ-менеджмент Ericsson решил не брать на себя обязательств по разработке и поддержке собственного языка программирования, сосредоточившись вместо этого на Java. Использование Erlang было запрещено в новых проектах Ericsson Radio AB в связи с реализацией плана по аутсорсингу программной технологии компании Rational Inc. Это решение очень сильно повлияло на будущее Erlang: оно привело к открытию кода Erlang под открытой лицензией EPL (аналог Mozilla Public License), а также послужило главной причиной начала распространения языка за пределами создавшей его компании. Основным возражением против открытия исходного кода являлось решение вопросов, касающихся патентов, но эти трудности были преодолены. Вскоре многие из основных разработчиков покинули Ericsson, чтобы организовать собственный бизнес — Bluetail AB.

В начале 2000-х учёные круги стали проявлять интерес к Erlang. С 2002 года стал проводиться ежегодный Erlang Workshop. Ericsson продолжал спонсирование проекта HiPE (от англ. High-Performance Erlang — высокопроизводительный Erlang) уппсальского университета. Проект HiPE занимался эффективной реализацией языка и инструментами для проверки типов, а с 2001 года созданный в группе проекта компилятор в машинный код входит в поставку свободно-распространяемой версии Erlang/OTP. Работы, связанные с Erlang, ведут и другие высшие учебные заведения. Инструменты для рефакторинга созданы в Кентском университете в Великобритании и университете Лоранда Этвёша в Венгрии, инструменты для различных видов тестирования — в Мадридском политехническом университете, техническом университете Чалмерса и Гётеборгском университете.

Когда системы с симметричной многопроцессорностью только начинали завоёвывать рынок серверов и настольных компьютеров, бросая вызов разработчикам программного обеспечения, уже в 2006 году первая версия Erlang с поддержкой SMP была выпущена совместными усилиями команды OTP из Ericsson и команды HiPE. Вскоре после этого вышла первая почти за десятилетие крупная монография по Erlang: «Programming Erlang» Джо Армстронга, после чего многие разработчики «открыли» для себя Erlang/OTP, и язык стал набирать популярность.

Процесс развития языка включает в себя рассмотрение предложений по развитию — EEP (англ. Erlang Enhancement Proposal). Через эти предложения Erlang-сообщество вносит изменения в стандартную поставку Erlang. С внесёнными предложениями можно ознакомиться на веб-странице erlang.org/eeps.

Философия

По свидетельству Майка Вильямса, Erlang задумывался для решения трёх проблем разработки распределённых систем мягкого реального времени с высокой степенью параллелизма: возможности быстрой и эффективной разработки ПО; получения системы, устойчивой к про-

граммным и аппаратным сбоям, и возможности обновления системы «на лету», без простоя оборудования.

По словам Вильямса, философия, которой придерживались разработчики Erlang, подходит и для разработки программного обеспечения на этом языке:

- Найдите наиболее подходящие методы. Проектируйте с использованием прототипов;
- Одних идей мало: нужно уметь их реализовать и знать, как они работают;
- Делайте ошибки в небольшом масштабе, а не в производственном проекте.

Большинство языков, созданных прежде Erlang, были разработаны без предварительного нахождения своей области применения, тогда как Erlang был разработан специально на основе требований к распределённым, отказоустойчивым, параллельным системам реального времени. С развитием сети Интернет оказалось, что многие приложения имеют аналогичные требования, чем и объясняется растущий интерес к языку.

Высокая отказоустойчивость кроется в применении изолированных друг от друга легковесных процессов, связанных лишь механизмом обмена сообщениями и сигналами выхода. Принцип разработчиков на Erlang по отношению к обработке ошибочных ситуаций в процессах можно выразить в виде высказывания:

> Позвольте приложению упасть, и пускай что-то другое имеет с этим дело.

или сокращённо — «let it crash» («пусть падает»). Связано это с тем, что в Erlang-системе легко следить за завершением процесса, завершать процессы, связанные со сбойным, и запускать новые процессы.

Основыне особенности

Высокоуровневые конструкции

Erlang является декларативным языком программирования, который скорее используется для описания того, что должно быть вычислено нежели как. Например, определение функции, которое использует сопоставление с образцом для выбора одного из вариантов вычисления или извлечения элемента данных из составной структуры, напоминает уравнение. Сопоставление с образцом распространено даже на битовые строки, что упрощает реализацию телекоммуникационных протоколов.

Функции являются объектами первого класса в Erlang. В языке также широко применяются характерные для функциональной парадигмы программирования списковые включения (генераторы списков).

Параллельные вычисления

Отличительной особенностью языка является применение легковесных процессов в соответствии с моделью акторов. Такой подход позволяет выполнять одновременно сотни тысяч и даже миллионы таких процессов, каждый из которых может иметь скромные требования по памяти. Процессы изолированы друг от друга и не имеют общего состояния, но между ними можно установить связь и получать сообщения об их состоянии. Для взаимодействия процессов используется асинхронный обмен сообщениями. Каждый процесс имеет

свою очередь сообщений, обработка которой использует сопоставление с образцом. Процесс, отправивший сообщение, не получает уведомления о доставке, даже если идентификатор процесса-получателя недействителен или получатель игнорирует сообщение. Таким образом, ответственность за правильно организованное взаимодействие между процессами лежит на разработчике.

Например, при реализация на Erlang сетевого чата структура программы может напрямую отражать одновременность действий пользователей по обмену сообщениями путём запуска новых процессов. Эффективность передачи сообщений сохраняется и при увеличении числа процессов, а требования к памяти минимизируются за счёт того, что легковесными процессами управляет виртуальная машина, а не средства нижележащей операционной системы.

Распределенные вычисления

Erlang с самого начала проектировался для распределённых вычислений и масштабируемости. Распределение вычислений встроено в синтаксис и семантику языка, поэтому построение системы можно вести, абстрагируясь от конкретного места вычислений. В стандартной поставке Erlang может наладить связь процессов по протоколу TCP/IP независимо от поддерживаемых им нижележащих платформ (операционных систем).

Работающий экземпляр среды выполнения Erlang (англ. Erlang runtime system) называется узлом (англ. node). Программы, написанные на Erlang, способны работать на нескольких узлах. Узлами могут быть процессоры, многие ядра одного процессора, и даже целый кластер машин. Узел имеет имя и «знает» о существовании других узлов на данной машине или в сети. Создание и взаимодействие процессов разных узлов не отличается от организации взаимодействия процессов внутри узла. Для создания процесса на другом узле процессу достаточно знать его имя и, без особых на то оснований, он может не интересоваться физическим расположением взаимодействующего с ним процесса. Синтаксис отправки сообщения процессу на своём узле и удалённом — один и тот же.

Благодаря встроенным в язык возможностям распределённых вычислений объединение в кластер, балансировка нагрузки, добавление узлов и серверов, повышение надёжности вызывают лишь небольшие затраты на дополнительный код. По умолчанию узлы спроектированы для работы внутри обособленного сегмента сети (DMZ), но, если необходимо, коммуникация между узлами может происходить с применением защищённого криптографическими методами протокола SSL.

Мягкое реальное время

Программы на высокоуровневом языке Erlang могут быть использованы в системах мягкого реального времени (которое иногда переводят как «псевдореальное» или «квазиреальное»). Автоматизированное управление памятью и сборка мусора действуют в рамках одного процесса, что даёт возможность создавать системы с миллисекундным временем отклика (даже несмотря на необходимость сборки мусора), не испытывающие ухудшения пропускной способности при высокой нагрузке.

Горячая замена кода

Для систем, которые не могут быть остановлены для обновления кода, Erlang предлагает горячую замену кода (англ. hot code upgrade). При этом в приложении могут одновременно

работать старая и новая версии кода. Таким способом программное обеспечение на Erlang может быть модернизировано без простоев, а выявленные ошибки исправлены.