Erlang Virtual Machine

Эликсир и Эрланг объединяет виртуальная машина EVM
(Erlang Virtual Machine),

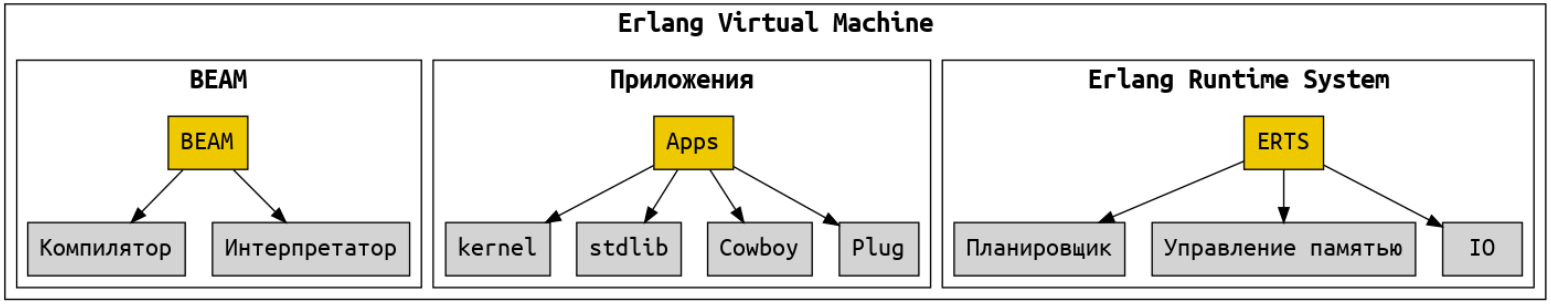
которую обычно называют **BEAM**
(Bogdan’s Erlang Abstract Machine).

Erlang Virtual Machine

Операционная система в миниатюре:

- планировщик процессов,
- управление памятью,
- ввод-вывод,
- сетевой стек,
- и др.

Erlang Virtual Machine



Erlang Virtual Machine

Основные свойства машины:

- Многопоточность (Concurrency);
- Устойчивость к ошибкам (Fault Tolerance);
- Поддержка распределенных систем (Distribution);
- Горячее обновление кода (Hot Code Upgrade).

Erlang Virual Machine

Еще несколько свойств:

- Симметричная многопроцессорность (Symmetric Multiprocessing);
- Модель акторов (Actor Model);
- Система реального времени (Soft Real Time);
- Сборщик мусора (Garbage Collector);
- Интерактивная консоль (Erlang/Elixir Shell);
- Трассировка (Tracing).

Многопоточность

Процессы (thread) и планировщики (scheduler), независимые от ОС.

Процессы легковесные, их можно создавать десятки и сотни тысяч.

Многопоточность

Каждый процесс имеет свою изолированную память, разделяемой памяти (shared memory) нет.

Процессы изолированы, блокировка или падение одного процесса не влияет на работу остальных.

Данные передаются отправкой сообщений (message passing).

Многопоточность

BEAM способна запускать до **134,217,727** (2^{27}) процессов.

Запуск нового процесса занимает **3-5 микросекунд**.

На старте процесс резервирует **2696 байт** памяти, включая стек, кучу и метаданные.

Многопоточность

Отдельный планировщик на каждом ядре CPU.

Балансируют нагрузку, передавая процессы друг другу.

Устойчивость к ошибкам

Исключения не основной способ обработки ошибок.

Основной способ – это механизм мониторинга
одних процессов другими.

supervisor (наблюдатель) и **worker** (рабочий процесс).

Устойчивость к ошибкам

Супервизоры наблюдают за рабочими процессами
и друг за другом.

Организованы в дерево, где узлами являются супервизоры,
а листьями – рабочие процессы.

Устойчивость к ошибкам

Следущий уровень устойчивости к ошибкам – объединение узлов в кластер.

Если узел падает, то его функцию берет на себя другой узел.

Поддержка распределенных систем

Устойчивость к аппаратным авариям.

Железо выходит из строя не часто,

но в дата центре из сотен серверов это случается регулярно

и является штатной ситуацией.

Поддержка распределенных систем

Горизонтальное масштабирование позволяет

строить большие системы, справляющиеся

с большими нагрузками и большими данными.

Поддержка распределенных систем

Сетевая прозрачность (location transparency).

Процессы на одном узле и на разных узлах

общаются одинаково.

Поддержка распределенных систем

Доверенная среду (trusted environment).

Любой процесс может посылать любые сообщения кому угодно.

Любой код выполняется с равными правами, без ограничений.

Горячее обновление кода

Важно для телекомуникационного оборудования 80-90-х годов.

Обновление без прерывая обслуживания клиентов,

без разрыва текущих телефонных сессий.

Горячее обновление кода

В современном мире мы умеем поддерживать работу кластера

при выходе из стоя части узлов.

Используем это для обновления.

Горячее обновление кода

Полезно как инструмент разработки.

Используем локально на машине разработчика.

Симметричная многопроцессорность

BEAM эффективно использует все CPU.

Умеет перераспределять нагрузку между ними.

Симметричная многопроцессорность

Производительность линейно масштабируется
с ростом числа ядер.

Проверяли на чипах с 1024 ядрами.

Работает ”из коробки”.

Модель акторов

Один из способов реализации многопоточности.

Модель акторов

Система состоит из акторов,

которые действуют паралельно и независимо друг от друга,

имеют собственное состояние

и общаются друг с другом с помощью отправки сообщений.

Модель акторов

Для модель акторов реализована как библиотека.

Например, библиотека **Akka** для Java и Scala.

Но в BEAM эта модель поддерживается на уровне языка.

Система реального времени

Реагирует на запросы с гарантированным лимитом времени.

Нарушение лимита времени считается
отказом системы (**hard real-time**),

или снижением качества работы системы (**soft real-time**).

Система реального времени

Применяется в промышленности, транспорте, медицине.

Управление атомной электростанцией.

Авионика самолета.

Беспилотный автомобиль.

Система реального времени

BEAM поддерживает **soft real-time** благодаря:

- вытесняющей многозадачности (preemptive scheduling);
- изолированному IO;
- особенностям сборки мусора (garbage collection).

Сборщик мусора

Отдельный для каждого процесса,

срабатывает независимо от других процессов,

в разные моменты времени.

Нет эффекта **stop world**.

Сборщик мусора

Вообще не запускается:

- короткоживущие процессы;
- долгоживущие процессы, потребляющие мало памяти.

Сборщик мусора

Сборка мусора оказывает мало влияния

на производительность системы.

Интерактивная консоль

REPL-консоль (**R**ead, **E**val, **P**rint, **L**oop).

Работает не только локально, но и удаленно.

Интроспекция – чтение информации о системе
в реальном времени.

Можно взаимодействовать с production системой.

Трассировка

Получение событий системы в реальном времени:

- вызовы функций, аргументы, возвращаемые значения;
- работа планировщика;
- потреблении памяти;
- работа сборщиков мусора;

Трассировка

Получение событий системы в реальном времени:

- жизненный цикл процессов (старт, остановка и др);
- отправка и получение сообщений;
- состояние памяти процессов.

Трассировка

Теоретически можно узнать почти все о работе системы.

Сложность в том, чтобы найти именно ту информацию, которая нужна.