rus.minimal_erlang_application_cluster

Alexandr Kirilov (https://github.com/alexandrkirilov)

Минимальная конфигурация кластера для распределенного Erlang приложения.

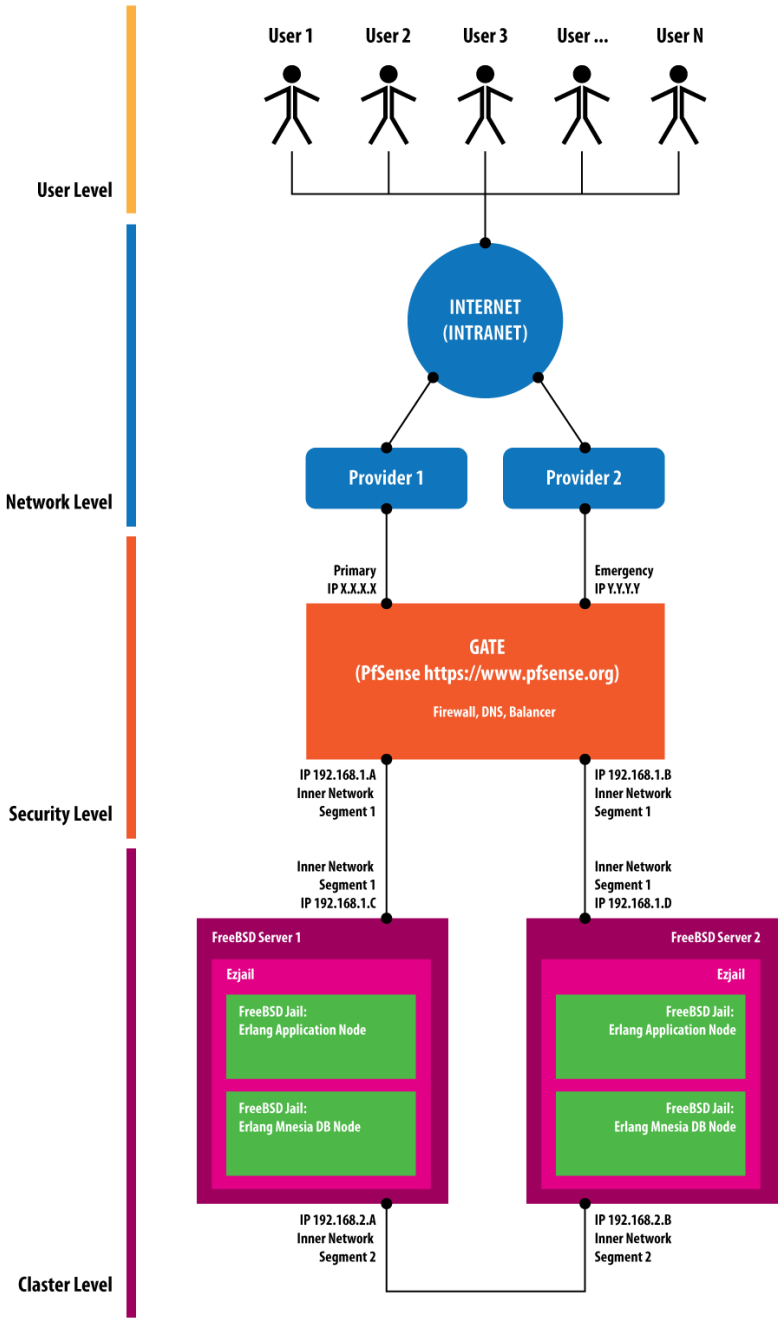
Данный пример конфигурации для распределенного Erlang приложения основан на реализованных проектах в различных сферах (сетевые игры, медицина, образование, платежные системы) когда бюджет на реализацию проекта был очень сильно ограничен. Конфигурация кластера описанная в этой статье является базовой и может быть использована с дополнениями или изменениями в соответсвии с приложением и ситуацией в которой она используется.

Задача: разработать отказоустойчивую архитектуру кластера для распределенного Erlang приложения обеспечивающую доступность приложения по схеме 24/7 с максимально возможной надежностью при ограниченном бюджете.

Для начала стоит разобраться в термине "распределенное приложение" (distributed application). Основной вопрос - почему ваше приложение спроектировано и разработано как распределённое? Ответы на этот вопрос определяют архитектуру кластера:

- Вам нужна высокая надежность и отказоустойчивость? - Да, решение: некоторая избыточность обеспечивающая надежность и отказоустойчивость при выходе из строя одного из элементов кластера.
- У вас возможны высокие нагрузки и один сервер может не обеспечить обслуживание всех пользователей? - Да, решение: горизонтальное масштабирование по типам узлов в кластере.
- У вас возможно хранение больших объемов данных? - Да, решение: автоматизированное клонирование хранилищ в случае необходимости с сохранением информации о том какие данные и где хранятся.
- И т.д.

Этот список отражает наиболее значимые причины для использования распределенной архитектуры приложения. Если ни один из этих пунктов не применим к решению вашей задаче - вам не нужно разрабатывать распределенное приложение. Относительно ваших задач это может оказаться не нужным и не оправданным по расходам.



Представленная схема описывает решение которое может удовлетворить все перечисленные выше требования небольшой компании или организации (торговая компания, образовательное учреждение, небольшой стартап, больница или какая либо

другая организация) намеренная развернуть сервера либо в дата центре, либо в помещениях где эта организация расположена.

Работоспособность данной архитектуры была проверена на проектах:

- REST API для мобильных, десктоп и web приложений способное выдержать +30K соединений в секунду (значение напрямую зависит от характера запросов со стороны пользователя, в некоторых случаях может больше а может быть и сильно меньше)
- Небольшое файловое хранилище
- Агрегатор событий системы безопасности
- Агрегатор статистики использования мобильных, десктоп и web приложений

Технологический стек этого решения состоит:

- **FreeBSD** (www.freebsd.org): базовая ОС для всех, причины обуславливающие выбор этой ОС как базовой описаны в статье "Почему Freebsd?"
- **PfSense** (www.pfsense.org): основано на FreeBSD, по сути это является самым бюджетным и массовым решением для обеспечения безопасности кластера в пределах любой компании, как маленькой так и большой. Автор статьи не использовал устройства PfSense, использовалась только сама операционная система установленная на относительно слабый сервер с некоторым количество сетевых интерфейсов. Для обеспечения контроля трафика мощный сервер не нужен. При грамотной настройке это может быть дешевым аналогом устройств Cisco с гораздо большей функциональностью.
- **FreeBSD Jails** (www.freebsd.org/doc/handbook/jails.html): технология позволяющая добиться подобия виртуализации при минимальных затратах и при минимальной квалификации специалистов обслуживающих систему. Изначально была разработана для увеличения уровня безопасности. В Linux-family операционных системах существует аналог, но практическое использование показало, то что это работает требуя больших ресурсов и медленнее. FreeBSD Jail может быть настроен таким образом что со стороны пользователя это будет выглядеть как отдельный сервер и вы можете установить таких Jail Server столько, сколько позволит вам железо на котором это все работает. При использовании связки Jenkins (jenkins.io) + BASH Script (Wiki Bash) + Ansible (www.ansible.com) вы можете добиться уровня автоматизации сходного с сервисами Amazon в пределах вашей собственной фирмы или организации не используя ни один сторонний сервис.
- **Ezjail** (Freebsd: Managing Jails with ezjail): сервисное приложение позволяющее организовать работу с Jails максимально простым способом. Если вам чем-то не понравилось это приложение, используя BASH Scripts вы можете написать свой

механизм для обслуживания Jails.

Эти четыре пункта описывают все что вам нужно для реализации этого маленького кластера, в архитектуру которого уже заложена возможность роста с самого начала. Поскольку со стороны пользователя FreeBSD Jails выглядят как отдельный сервер и со стороны администрирования они выглядят как настройка отдельного сервера, то при росте нагрузки вы совершенное спокойно можете добавить сервер на который вы сможете масштабировать ваше приложение, даже в автоматическом режиме если захотите.

Стоит отдельно отметить то, что внутри Erlang приложения происходит разделение уровня данных от уровня приложения, что обеспечивает дополнительную безопасность. Приложение расположено в отдельном jail и сервер баз данных расположен в отдельном jail. Логически это выглядит как отдельные сервера в сети.

Основой этой архитектуры является понимание того, что вы рано или поздно будете расширяться и вам придется масштабировать свое приложение и зная это вы заранее заглядываете эту возможность:

- специально разделяете данные от приложения и если возникла нагрузка переносите jail на физический сервер а соответственно запросы к базе данных вы реализуете будто это отдельный сервер
- специально организуете внутренний DNS который позволит балансировать внутреннюю нагрузку
- и т.д.