敏行平台系统扩展手册

发布日期：2014 年 9 月

版本：V3.0

目　　录

1 引言

1.1. 编写目的

敏行平台在最初的设计上就考虑到为未来系统的容量扩展提供架构上的支持，该文档描述了如何对敏行的架构进行扩展，以便使敏行平台获得更好的性能，通过敏行系统架构的扩展，使敏行系统服务更多的用户数。对于 10 万用户内的环境，都可以按照本文档的处理方式增强系统的处理能力，超出 10 万用户则需要对架构进行重新的设计考虑。

1.2. 系统概述

敏行默认的系统提供两种部署模式：

1. 单机运行环境，由一台服务器提供全部的服务。不建议在生产环境下使用。
2. 双服务器环境，系统运行的主要服务做了负载均衡和 HA 的主备配置，可以提供生产环境下的高可靠性。

两种部署模式都可以进行系统的扩展来提高系统的负载能力。对于单机运行环境，可以将系统升级为双机运行的部署方式。已有的双机运行可以将部分的服务迁移出双机环境，以增强系统的处理能力。

2 系统架构

2.1. 软件组成

敏行平台的基础软件服务由下列软件构成：

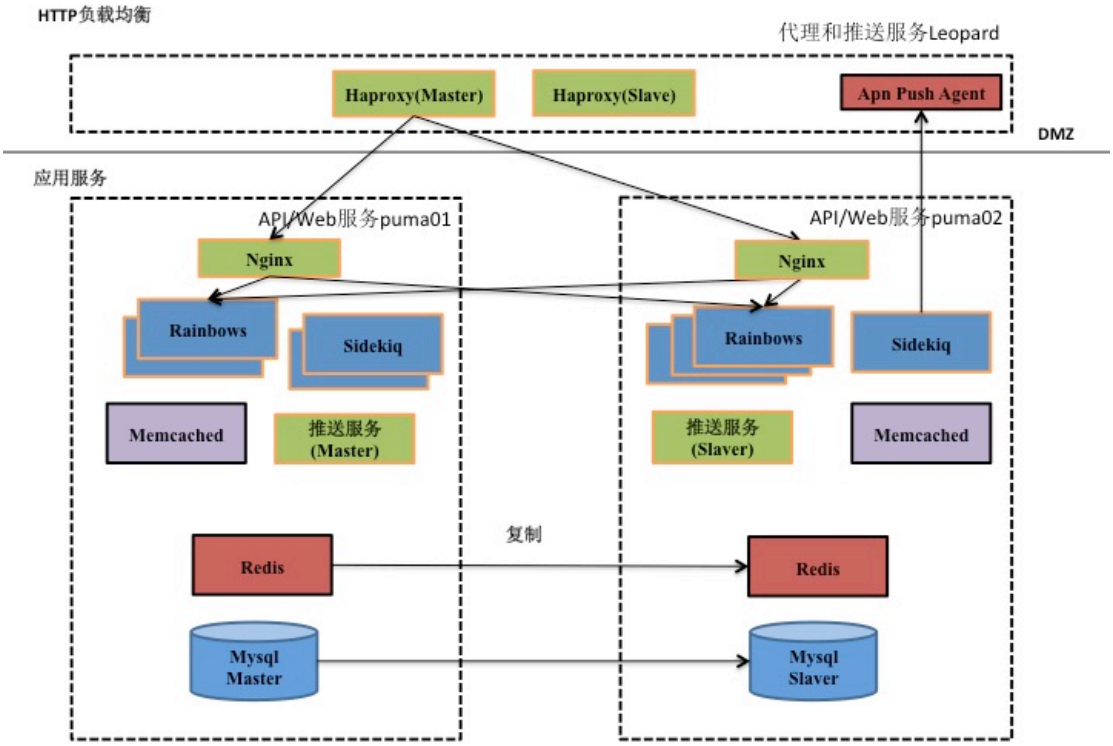
- 操作系统 Debain Linux 6.0.10 版本
- Web 服务器 Nginx 1.4.7
- 数据库 MariaDB 5.5.37
- 数据缓存 Redis 2.6.16
- 数据缓存 Memcached 1.4.5

- 应用服务器 Rails 3.2.19
- 推送服务 Nodejs 0.10.26

在单机环境下部署时，全部软件都运行在同一服务器内。对于双机的环境，这些服务会分布在两个服务器上。

2.2. 网络拓扑

系统双机环境下的网络拓扑架构如下



- 系统最外层可以由 Haproxy 作为负载均衡器，将 web 层的网络请求分发到两台服务器上。
- 应用服务器使用 Rainbows 作为处理的服务器。
- 系统中消息的分发和异步处理由 Sidekiq 的服务完成，异步处理的消息存储在 Redis 内，两台服务器中的 Sidekiq 都会监控 Redis 中出现的新消息。
- 敏行产生的消息存储在 Mysql 的数据库服务器中，Mysql 两台主机按照主从复制保持系统的可靠性。
- 系统使用分布式的 Memcached 来缓存系统数据，两台服务的 Memcachd 中各存储了一部分数据。

3 扩展方案

3.1. 单服务器扩展

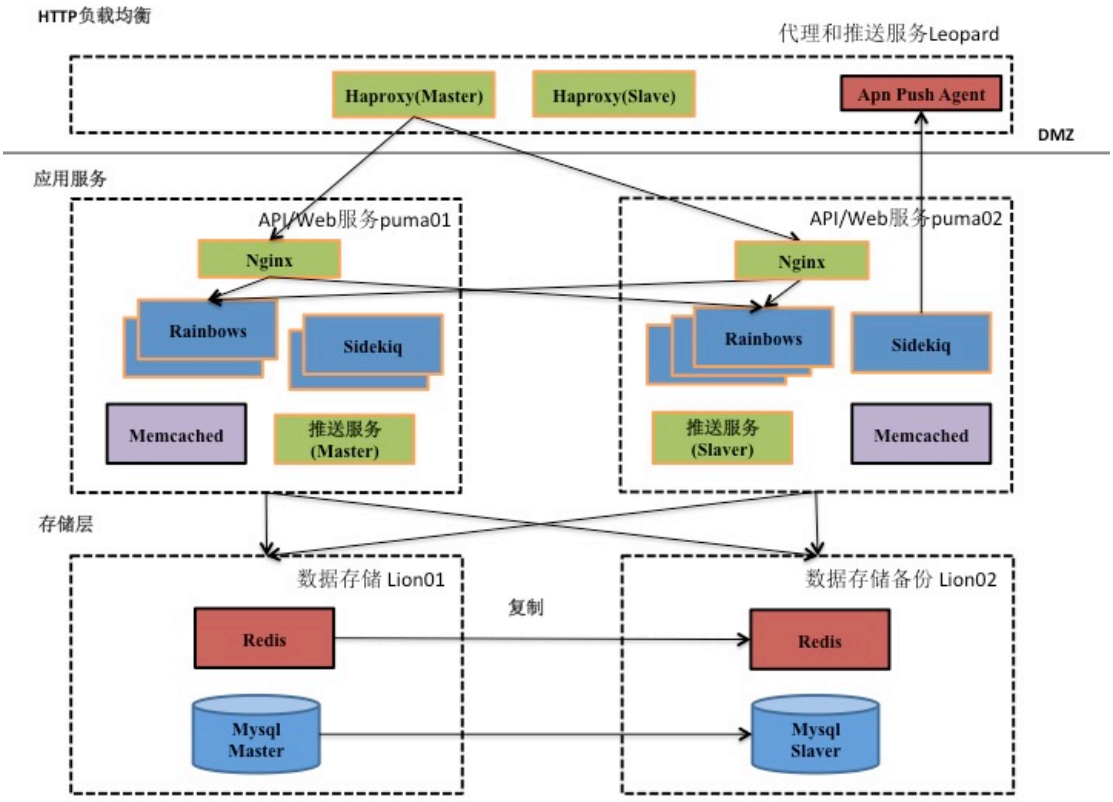
单服务器可以通过在硬件上增加更多的 CPU 和更大的内存来提高系统的处理能力，建议增加 CPU 的同时对内存进行响应的增加，同时还需要对系统进行下列配置。

- 增加 Rainbows 的进程数量，修改配置文件 /home/ewhine/deploy/ewhine_NB/config/rainbows.rb，将 worker_processes 的数量增加，建议这个值设置为 CPU 个数减一。例如，4CPU，则设置为 3。
- 增加 Sidkekiq 进程的运行数量，因为 Sidekiq 是 ruby 语言实现的，但目前 Ruby 的进程并不能真正利用系统的多 CPU 进行并行处理。
- 监测 memcached 的 cache 命中率，如果命中率过低，可以增加 memcached 的内存大小，提高系统的运行效率。

如果单台服务器的架构仍然不能满足系统用户访问量的要求，可以将单台服务器的架构改为双机的服务器架构，改用该架构不但能够给服务带来更好的性能，同时也能提高服务的可靠性。

3.2. 数据库扩展

在双服务器的架构下，系统如果性能仍然存在响应过慢的问题，那么这些请求很可能是被服务器的性能所拖慢的，可以将敏行平台的数据库服务使用单独的服务器进行部署，新的架构图如下：



将数据库服务器独立出敏行系统会有效的改善数据的存储效率：

- 数据库服务器有自己独立的内存和 CPU 资源，不会与其他的服务在资源上进行相互争用。
- 数据库系统独立服务器之后可以为数据库服务配置读写分离的策略，增加读请求的能力。

3.3. Sidekiq 扩展

敏行中的 Sidekiq 进程负责异步处理用户发送的消息，这些异步处理工作包括：

- 将消息存储到数据库中。
- 将用户消息推送到推送服务器中。
- 对 iOS 的客户端，还需要进行 apn 消息的推送。
- 对于 web 在线的用户，发送推送消息给客户端

Sidekiq 的服务监控 Redis 服务内的消息，一旦前段将消息写入 Redis 中，Sidekiq 则会将消息取出执行相应的操作，由于 Sidekiq 是纯 ruby 实现的队列处

理程序，因此在 ruby 2.1.2 的版本下，还不能充分利用多核 cpu 的处理能力，为了扩展 Sidekiq 的处理能力，需要在系统中同时运行多个 Sidekiq 的实例。增加 Sidekiq 实例的同时，需要增加相应的内存配置。在一台 8 核的服务器上，建议配置的 Sidekiq 数量不超过 4 个。

3.4. Redis 扩展

敏行系统大量的使用了 Redis 的服务器做为数据的快速存储库，但 Redis 是一个单线程的服务，无法利用系统的多颗 CPU，通常情况下，Redis 的访问速度都比较快。但随着系统用户量的增加，Redis 可能会出现负载过高的问题。对 Redis 扩展可以使用 twemproxy 作为 Redis 的代理，将数据内容分布在 twemproxy 的后端。Twemproxy，是 Twitter 开源出来的 Redis 和 Memcached 代理。而 Twemproxy 通过引入一个代理层，可以将其后端的多台 Redis 或 Memcached 实例进行统一管理与分配，使应用程序只需要在 Twemproxy 上进行操作，而不用关心后面具体有多少个真实的 Redis 或 Memcached 存储。通过对 Redis 的扩展，敏行系统中的工作圈时间线和即使通信部分的信息都会得到比较好的性能扩展。

3.5. Web 层扩展

敏行系统的 web 端是主要有 Nginx 和 Rainbows 的群集构成，这部分是系统最容易扩展的部分。可以利用 Haproxy 建立对多台应用服务器的负载均衡。由于敏行使用了 cookie 做为用户会话保持的机制，服务器并不存储用户的状态数据，因此比较容易为 Web 服务的群集增加服务器或者减少服务器。利用虚拟话技术，可以为白天较多的请求动态增加 Rainbows 的服务器，而在夜间则减少 Web 服务的数量。

4 结论

敏行系统在最初的设计中就考虑过如何对系统进行快速的扩展，通过将敏行系统的服务单独拆离，可以快速提高服务的可用性，性能。并且能够在不做架构

重大调整的情况下，支持更多的用户。