



Dingo

Dindo Document

DingoLab
May, 2016

Dingo Dindo Document

李约瀚

qinka@live.com

14130140331

May 23rd 2016

Version 0.0.1.0

西安, Xi'an

DingoLab

前言

这个文档是 Dingo 后端 Dindo 的文档，包括后端的大体需求说明，宏观设计说明、详细设计说明、数据库设计与实现、软件源码说明、软件测试说明、软件部署说明件与软件使用说明。

后端 Dindo 使用 Haskell ¹，与 Yesod 框架 ² 编写的。同时整个后端代码中 Haskell 的部分是使用 Haskell 与 L^AT_EX 混排的文学编程。所以文档中有一部分为程序代码（及其说明）。

Dindo 的名称由来是在笔者（也是主要维护者）在数学建模的校赛是，使用 Lingo 是受到 Lingo 与 Lindo 的关系而起的名字。

这个后端依次将介绍需求、设计、数据库设计、软件部署、软件使用与维护、Dindo 代码及其说明等内容，以上是正文部分。附录中将会有部分术语表、维护的文档、索引、参考文档等。

¹Haskell 是一门纯函数式的编程语言。

²Yesod 是一个使用 Haskell 作为主要语言，的 RESTful API 的 WEB 应用框架。

目录

| | |
|----------------------|---|
| 1 大体需求说明 | 1 |
| 2 Dindo 架构设计概论 | 1 |
| 3 均衡负载设计 | 1 |
| 4 弹性计算设计 | 2 |
| 5 微服务架构设计 | 2 |
| 6 业务流程说明 | 2 |
| 7 数据库设计 | 2 |
| 8 Dindo 部署说明 | 2 |
| 8.1 测试部署方式 | 2 |
| 8.1.1 原生运行 | 2 |
| 9 Dindo 软件使用与维护说明 | 3 |
| 10 Dindo 源码及说明 | 3 |
| 11 Dindo 公共组件 | 3 |
| 12 Dindo 数据库 | 3 |
| 13 Dindo Launcher | 3 |
| 14 Dindo 微服务组件——用户管理 | 3 |
| 15 DIndo 测试说明 | 3 |
| A 术语解释 | 4 |
| B Docker 中 Weave 的配置 | 4 |

1 大体需求说明

2 Dindo 架构设计概论

Dindo 是 Dingo 的核心部分之一，负责客户端与后端的交互，同时负责客户端与数据库的、客户端之间的间接交互。此部分将有负载均衡的大致方法、弹性计算的解决方案、后端 API 与服务程序分割的内容。同时还将说明后端业务流程。

Dindo 是基于 Docker 容器上，采用微服务架构的一个后端。所有的组件将运行与 Docker 容器之中，且方便运行与公有云搭建的 Docker 中，同时价格相对比较便宜。按照灵雀云的收费标准 [1]，按照北京一区（AWS）来计算。当不使用弹性计算中的策略，即仅当容器的大小与数量时确定不变时。负载均衡负载的采用一个 M 级别的容器，运行 5 个 L 级别的容器作为数据库，运行 20 个的 M 级别的容器为处理业务的核心部分。数据库每个容器配置 100G 的挂载点用于存放数据，并计划每天下载数据量有 10G。按上述配置需要³

$$((20 + 1) * 0.329 + 5 * 0.658) * 24 * 30 + 10 * 30 * 0.93 + 0.75 * 100 * 5 = 7997.28$$

每个月大致需要不到 8000 元的成本⁴。

Dindo 开发过程依赖敏捷开发，并采用以持续集成为主的测试方式测试，同时采用持续交付的方式交付运营者。由于采用微服务架构、持续交付与 Docker 可以使得后端的版本升级处于“无痛”状态。微服务架构也能使的后端的业务逻辑分布在不同的程序（组件），也可使得后端分布上线。

3 均衡负载设计

均衡负载采用 Nginx 作负载均衡的软件，

³一个月按 30 天计算。

⁴当采用弹性计算时，这个成本将继续下降

4 弹性计算设计

5 微服务架构设计

6 业务流程说明

业务流程部分包括后端对事件驱动型的业务处理过程，每个 API 中业务处理过程等。这部分的主要内容将在 Dindo 源码及其结束的部分说明。

7 数据库设计

8 Dindo 部署说明

此部分主要说明 Dindo 的部署问题，包括测试、原型与最后实际运行是的部署。测试与原型的部署有两种方式，一种是直接运行，另一种是基于 Docker⁵。而最后运营是的部署，目前计划直接部公有云之上，利用 CaaS 服务。

8.1 测试部署方式

测试的部署一般适用于调试与检测。调试一方面是指后端开发时测试验证，另一方面则是指前端开发时测试使用。检测是如安全性测试等方面的检测。而通常运营部署通常不需要调试磨合，直接部署到 CaaS 提供商即可。

8.1.1 原生运行

原生运行首先要构建⁶然后部署，最后运行。如果已获得构建好的二进制文件，请直接跳过下面构建的过程。

Windows 下构建 首先需要安装 <https://www.haskell.org/platform/windows.html>，然后克隆 <https://github.com/DingoLab/DingoBackend> 仓库到本地，然后安装 stack，安装方式可参考 https://github.com/commercialhaskell/stack/blob/master/doc/install_and_upgrade.md#windows。安装完之后跳转到仓库的目录：

```
1 || cd DingoBackend
```

⁵基于的是 Ubuntu (Linux) 原声的 Docker，暂不讨论 Mac OS X 与 Windows 下原生的 Docker。

⁶Dindo 是不直接发行二进制文件的，发行的只有 Docker 镜像。

然后执行构建：

1

```
stack build
```

然后在 `.stack_work` 文件夹中某个文件夹下面的 `bin` 文件夹中可以找到编译好的二进制文件⁷。

Linux 下的构建 首先安装 GHC⁸。安装的方式通常通过

9 Dindo 软件使用与维护说明

10 Dindo 源码及说明

11 Dindo 公共组件

这部分是关于 Dindo 的公共组件的。由于 Dingo 后端采用的微服务架构⁹，不同的微服务之间，会有包括服务发现¹⁰、数据库¹¹、授权认证等是共用的。所以为了减少代码的重复使用，则独立出这一部分。

12 Dindo 数据库

13 Dindo Launcher

14 Dindo 微服务组件——用户管理

15 Dindo 测试说明

⁷为何不直接搜索。

⁸要求 7.10 以上，之前的版本没有测试过，无法保证可以正常编译运行。

⁹后面随时可能会称之为微架构。

¹⁰目前的版本并没有开发实际的服务发现的内容，直接使用 Nginx 进行做均衡负载等。

¹¹这一部分单独出来的。

A 术语解释

CaaS Container as a Server，是指将容器（Docker）提供作为一种服务。是云计算中的概念，与 PaaS、SaaS 等概念对等。

B Docker 中 Weave 的配置

Weave 是能将 Docker 中每个物理主机中的连接起来一个工具，也就是能使用的 Docker 容器跨主机互联。下面是配置（安装）Weave 的 Shell 脚本：

Listing 1: Weave 安装

```
1 #!/bin/sh
2 wget -O /usr/local/bin/weave \
3 https://github.com/zettio/weave/releases/download/latest_release/weave
4 chmod a+x /usr/local/bin/weave
5 dao pull weaveworks/weave:1.5.1
6 dao pull weaveworks/plugin:1.5.1
7 dao pull weaveworks/weaveexec:1.5.1
8 apt-get update
9 apt-get install bridge-utils
10 dao pull weaveworks/weavedb:latest
11 weave launch 192.168.1.181
```

运行容器需要使用

```
1 # weave run <ip> <repo>
```

参考文献

[1] 灵雀云收费标准 2016 年 5 月，[Alauda-Price](#)