



股票代码：000971.SZ

# 容器技术在CDN平台的实践

高升控股 2016年11月

# 高升控股企业架构

股票代码: 000971.SZ

21分支机构, 500+员工

在云基础服务领域历经10年耕耘

云管端垂直一体化

提供IDC、VPN、CDN、APM  
及相关增值服务



# 高升控股业务集群



## APM

云评测-云监测-云加速一站式服务  
已服务网站**36,868+**  
注册用户近**90,000**  
全球监测**882,050,000**次

## CDN

国内：**200+**节点，**2,000G+**带宽  
海外：已覆盖北美、欧洲、澳洲、  
东南亚、港澳台、日本等，计划开  
通印度、以色列、南非、西班牙、  
阿联酋等新节点

## 领先的云基础服务提供商

一站式、大带宽、高性价比  
全国**90+**城市**200+**星级机房  
**5,000G**带宽储备  
北京大规模IDC群落

因特网虚拟专用网经营许可证  
国内固定网数据传送业务经营许可证  
专用网业务节点覆盖**90+**城市

## IDC

## VPN

[www.gosun.com](http://www.gosun.com)

# 目录

## CONTENTS



### CDN平台业务问题



### 容器化带来的解决方案



### 系统设计要点

[www.gosun.com](http://www.gosun.com)



# CDN后台业务问题

## 整体架构



# CDN后台业务问题

## CDN后台业务系统包括

1. 分发系统 2. 资源管理系统 3. 分布式监控系统 4. GSLB调度系统 5. GSVC后台管理系统

1. nginx 2. mysql 3. mongodb 4. redis 5. tomcat 6. resin 7. uwsgi 8. gunicorn

## 这些业务用到的服务

## 运行业务程序需要的环境

1. python2/3 2. java 3. Ruby 4. C 5. node

# CDN后台业务问题

将整个后台业务系统拆分为多个相对独立的应用，有以下特点：

1

模块可以独立提供服务，边界清晰、易于维护。

2

可以用不同语言编写，易于引入新技术。

3

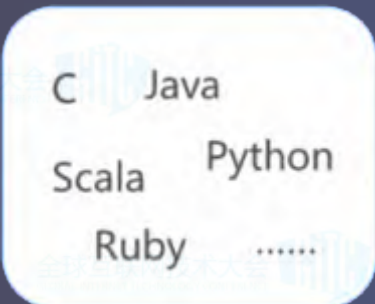
模块间松耦合，不同 SLA 保障计划。

4

更好的可扩展性和鲁棒性

## CDN后台业务问题

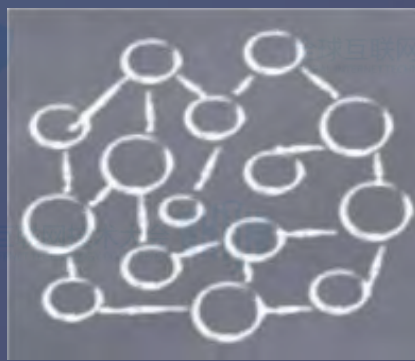
### 拥有特点的同时带来的问题



使用不同的语言开发应用

- 需要为每种语言准备编译环境
- 需要为每个服务准备不同的支持库和框架
- 种类一多，开发复杂、部署麻烦

- 复杂的版本管理和BUG跟踪间接导致项目管理成本增加。
- 每个程序的分布式需要自行解决服务注册等问题。



大量的微服务模块

# 容器化带来的解决方案

## 资源独立、隔离



不同应用以“集装箱”（container）为单位装“船”（主机集群），数千数万个“集装箱”排列整齐，不同公司、不同种类的“货物”（应用所需的程序，组件，环境）保持独立。



工程师完成应用开发后build一个 image，基于这个 image 创建的 container 像是一个集装箱，里面打包了各种“散件货物”（应用所需的程序，组件，环境）。无论这个集装箱在哪里：开发环境、测试环境、生产环境，都可以确保集装箱里面的“货物”种类与个数完全相同。每一个环节都是在运输这个完整的、不需要拆分合并的“集装箱”。

## 环境的一致性

## 轻量化



相比传统的虚拟化技术（VM），容器的性能损耗更低。可以在秒级启动与销毁。



“货物”（应用）在“汽车”，“火车”，“轮船”（私有云、公有云等服务）之间迁移交换时，只需要迁移符合标准规格和装卸方式的“集装箱”（container），削减了耗时费力的人工“装卸”（上线、下线应用），带来的是巨大的时间人力成本节约。

## Build Once, Run Everywhere



## 容器化带来的解决方案



- 容器化的联合作业能力？
- 应用程序对于容器的兼容？
- 开发人员额外的学习成本？
- 系统升级对业务的影响？



## 容器化带来的解决方案



## 容器化带来的解决方案

### Eagle

业务发布，版本管理，自动缩/扩容  
权限管理，监控，持续集成

### REKM

提供平台需要的数据存储，检索，消息队列等基础功能。

### Swarm

解决基于资源的分布式调度问题

### Docker

容器化，资源隔离，保持环境统一

EagleCloud

[www.gosun.com](http://www.gosun.com)

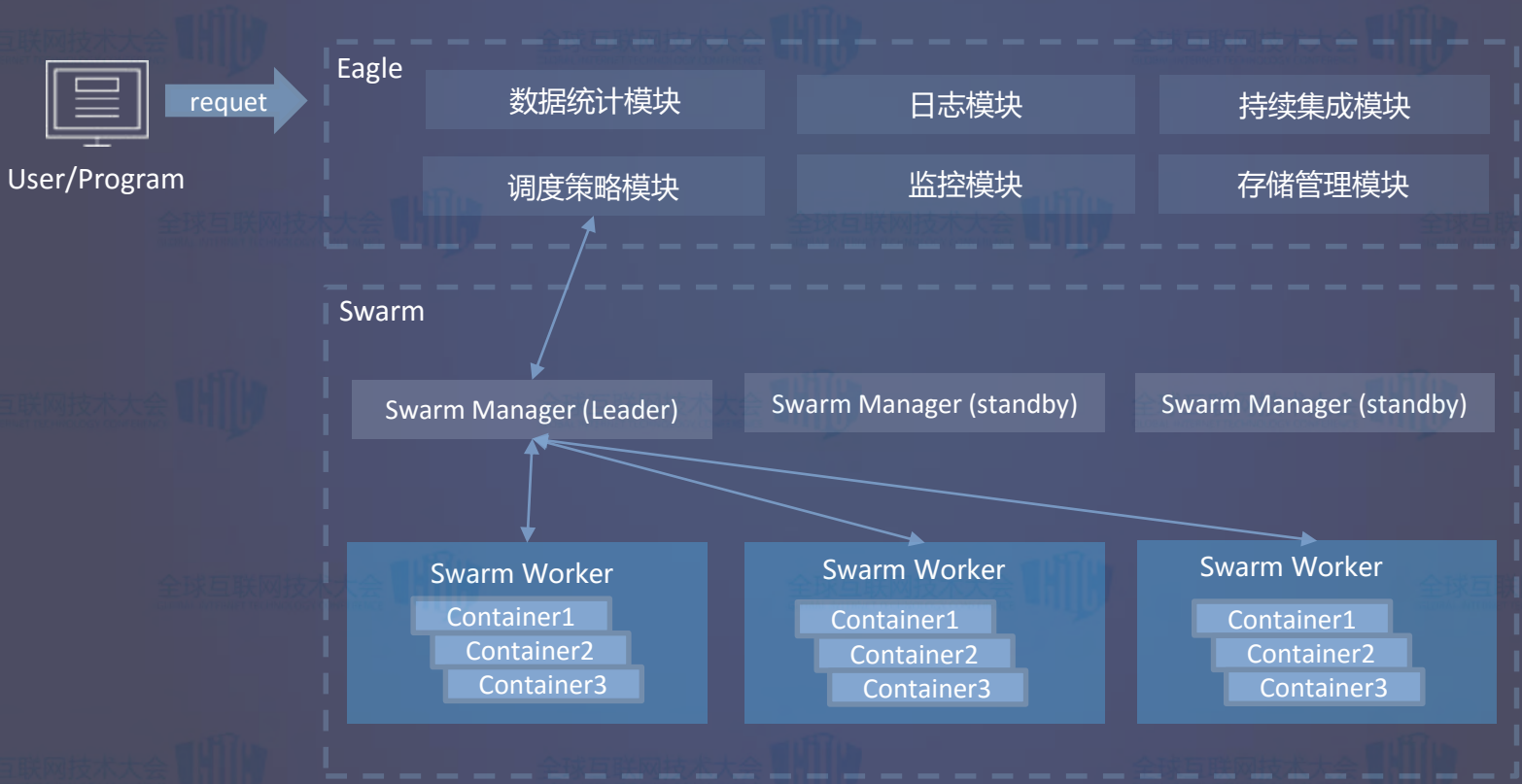
# 容器化带来的解决方案

基于Swarm实现，使用Service概念来定义应用，每个Service由多个容器组成。



# 系统设计要点

## 系统介绍



# 系统设计要点

## 持续集成



30%

### 代码获取

通过Webhook，从GitHub或GitLab中获得代码，开始构建



50%

### 自动构建

根据代码目录下的Dockerfile文件或系统中选定的配置模板在测试节点中构建镜像。



80%

### 镜像测试

针对构建的镜像启动测试容器进行应用测试



100%

### 镜像集成

将测试通过的镜像存入相应的镜像仓库，并记录相关信息。

# 系统设计要点

## 图形化的部署流程

运行环境 基本设置 高级设置

所属镜像 hub.gosun.cloud/library/nginx:latest

应用名称 test

应用组别 default

容器配置

| 配置 | 1核    | 1核    | 2核    | 4核     | 8核     |
|----|-------|-------|-------|--------|--------|
| 内存 | 1G/内存 | 4G/内存 | 8G/内存 | 16G/内存 | 64G/内存 |

数据中心 shanghai

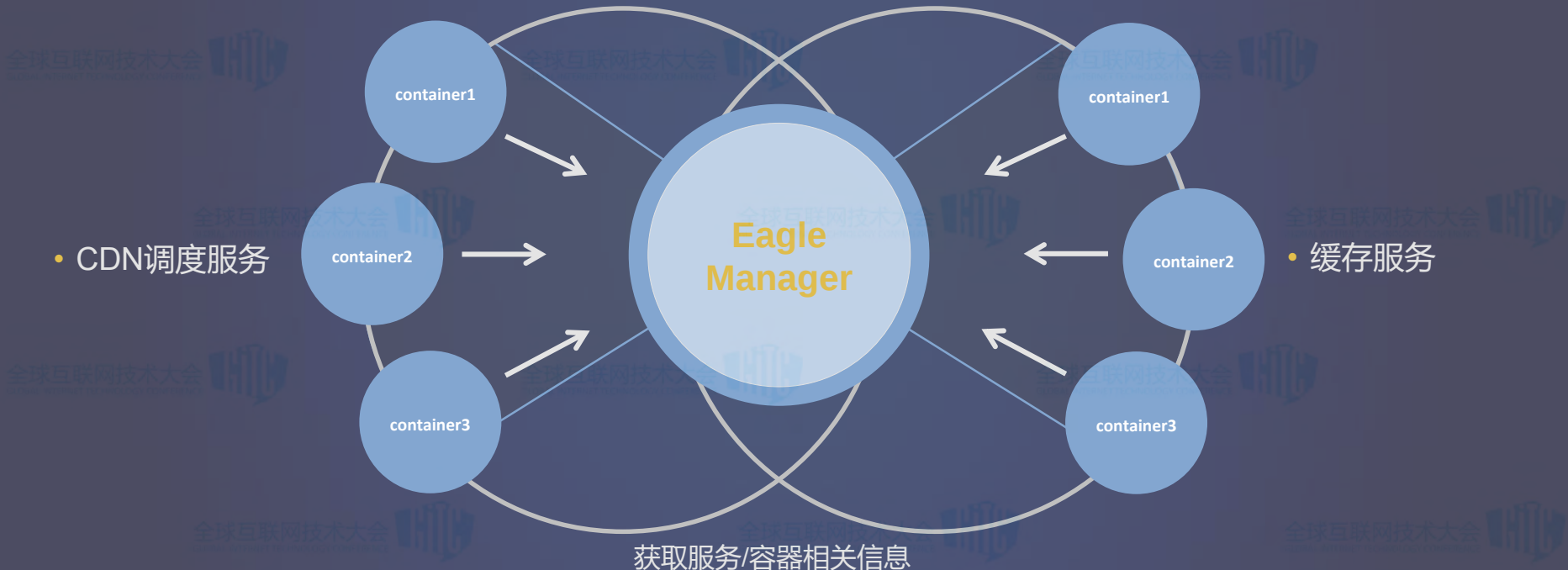
访问地址 122.228.212.134

容器个数 6

基础设置 >

# 系统设计要点

平台辅助程序注册



• 反馈JOSN事例

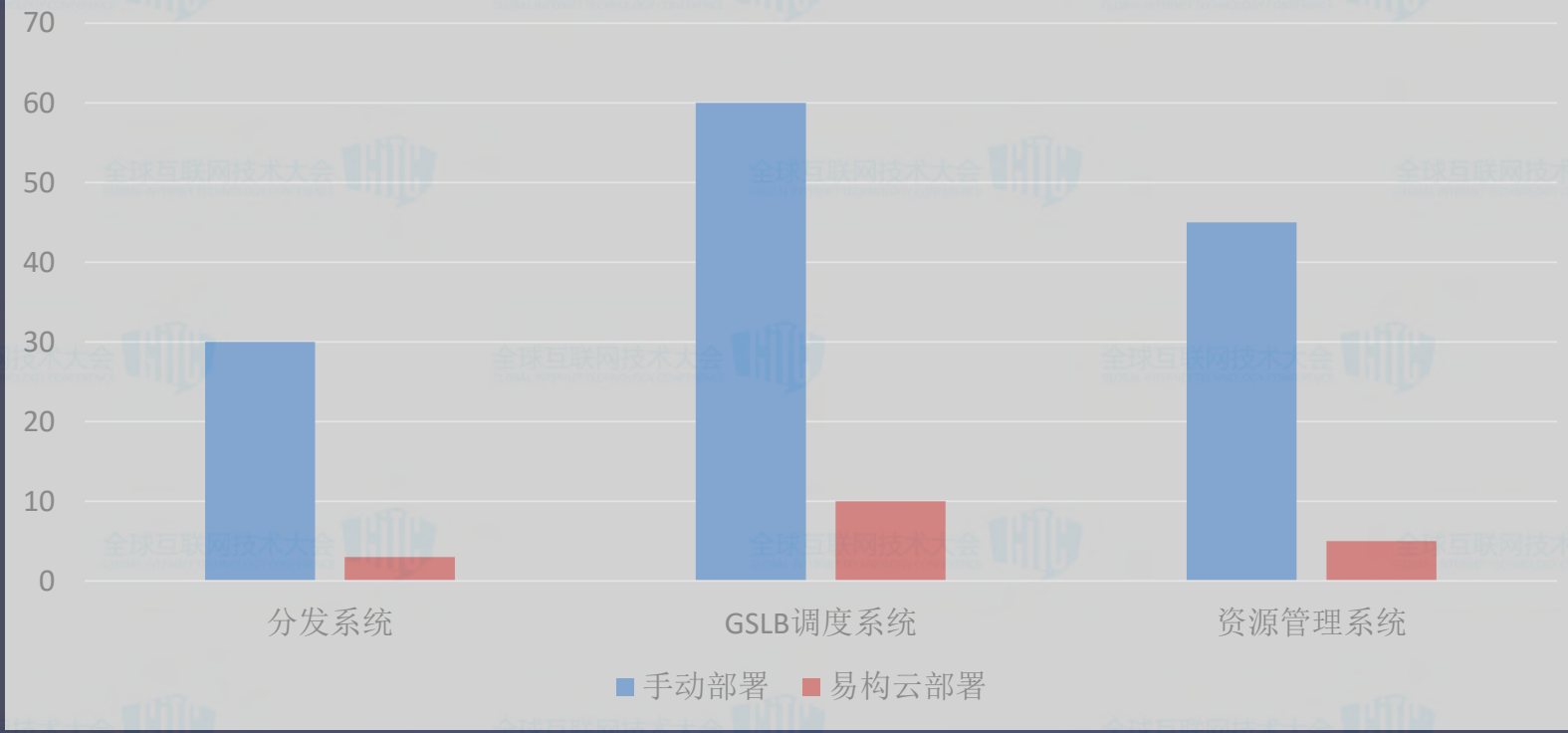
```
[{"ID":"73p2mb4zcql6dznrxz26zps63","Version":{"Index":477},"CreatedAt":"2016-11-18T09:32:13.545697905Z","UpdatedAt":"2016-11-18T09:32:13.627846278Z","Spec":{"Name":"cache","Labels":{"type":"slave"},"TaskTemplate":{"ContainerSpec":{"Image":"nginx"},"Resources":{"Limits":{"NanoCPUs":1000000000,"MemoryBytes":536870912},"Reservations":{"NanoCPUs":1000000000,"MemoryBytes":536870912},"RestartPolicy":{"Condition":"any","MaxAttempts":0},"Placement":{"Constraints":["node.role == manager"]},"Mode":{"Replicated":{"Replicas":3},"UpdateConfig":{"Parallelism":1,"FailureAction":"pause"},"EndpointSpec":{"Mode":"vip","Ports":{"Protocol":"tcp","TargetPort":80,"PublishedPort":80}}},"Endpoint":{"Spec":{"Mode":"vip","Ports":{"Protocol":"tcp","TargetPort":80,"PublishedPort":80}}},"VirtualIPs":{"NetworkID":"1ojn77lpo1safxepovhcib36i","Addr":"10.255.0.5/16"},"UpdateStatus":{"StartedAt":"0001-01-01T00:00:00Z","CompletedAt":"0001-01-01T00:00:00Z"}}
```



# 系统设计要点

业务部署时间对比

单位（分钟）



# 系统设计要点

## 网络

隧道技术 VS 路由技术

CNM VS CNI



# 系统设计要点

## Linux Bridge

适应简单的网络模型，隔离简单（MAC VLAN），对于业务来说稳定且性能不损失

## OVS Bridge

可以实现基于VLAN和VxLAN的网络，但VLAN不如Linux Bridge简单稳定，Vxlan也不如原生overlay方便。但可以实现基于openflow实现链路的负载均衡和流量统计。

后自带的网络模型，在1.12中与内置的swarm结合提供了完整的网络方案。稳定性比之前加强了不少但基于VxLAN还是有最高30%的性能损失。在对流量需求不大且隔离性较高的业务中比较合适

对于新版本的支持比较差，swarm-mode下暂时不支持1.13可能会改善  
<https://github.com/docker/docker/issues/24569>

## overlay docker

1.9

## calico weave flannel

# 系统设计要点

## 有状态服务

所有有状态的业务，通过docker自带的Volume机制存放数据。

高IO接入本磁盘，低IO接入大数据平台的分布式文件系统（GlusterFS）。

日志全部分离通过syslog送入大数据平台（Spark+kafka+elasticsearch+GlusterFS），容器内只需要将日志输出至stdout.

Volume plugin接口机制让对接后台的分部署文件系统异常容易。现有的plugin参考Rancher Convoy项目，已经支持的后台存储有Device Mapper, NFS, EBS, GlusterFS。并实现了备份，快照等扩展功能。



# 系统设计要点

## Swarm的问题

- 1: 1.12版本的Docker Swarm BUG众多，诸如在Iptables下内部转发策略缺漏， Overlay Network无法通过接口创建等。
- 2: 以前觉得Marathon的Event Callback够烂的了,Docker能比它还烂。
- 3: Docker Master 不转发其他节点消息，获取 指定 Container 信息要先查 node id,然后单独连接相关 node 获取信息，便捷性几乎没有。
- 4: 相关命令行和API的功能是不一定匹配的，比如scale是命令行才有的特权,api依旧要update传docker参数全家桶。
- 5: 官方Docker-py的文档遍地地雷,与Docker api对不上,甚至影响到了他们自身的Docker-compose项目。
- 6: Swarm的覆盖式设计理念完全不为分层架构考虑。





# THANK YOU



WWW.GOSUN.COM



4008 365 500