基于容器的分布式系统的设计模式

Design Patterns for container-based distributed systems

Brendan Burns & David Oppenheimer Google

Team: 孟慧玲 刘作禹 唐俊

Agenda

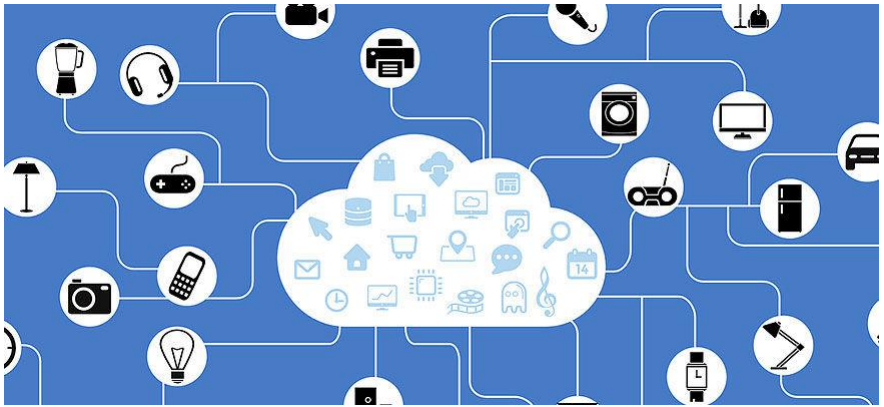
- Prequisite
  - Distributed System
  - Container
- Three Design Patterns
  - Single-container management patterns
  - Single-node, multi-container application patterns
  - Multi-node application patterns
- Related Work
- Conclusion & Storming
- Reference
- Q & A

Prequisite

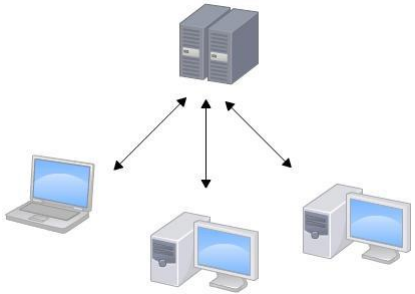
分布式系统?

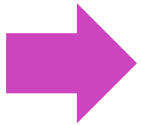
分布式系统是一个硬件或软件组件分布在不同的网络计算机上，彼此之间仅仅通过消息传递进行通信和协调的系统

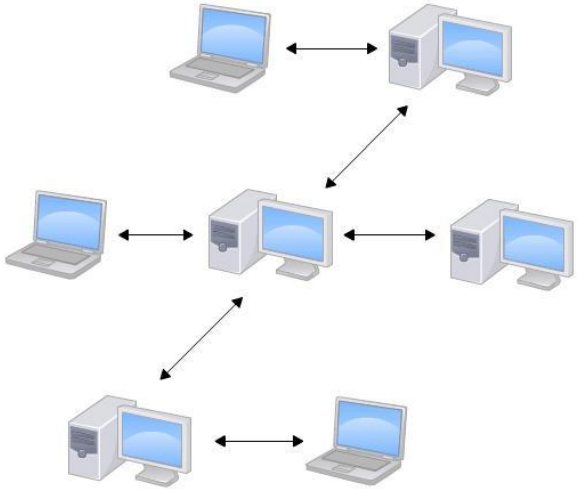
- 多设备多节点
- 分布式存储
- 并行计算
- 全局通信



集中式系统 VS 分布式系统







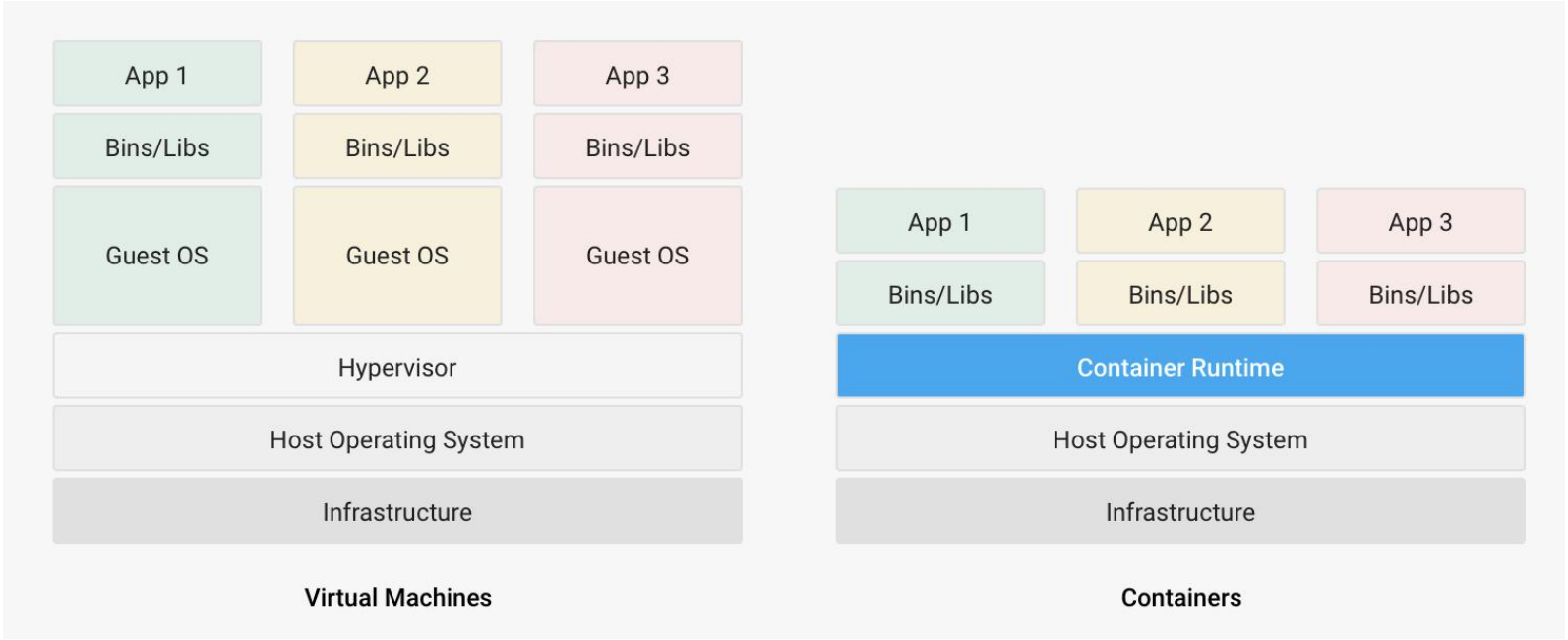
- 功能集中
- 程序集中
- 集中对外提供服务

- 分布存储
- 并行计算
- 全局通信

A quick Ques

How?

虚拟化 VS 容器化


App 1
App 2
App 3
Bins/Libs
Bins/Libs
Bins/Libs
Guest OS
Guest OS
Guest OS
Hypervisor
Host Operating System
Infrastructure
Virtual Machines
App 1
App 2
App 3
Bins/Libs
Bins/Libs
Bins/Libs
Container Runtime
Host Operating System
Infrastructure
Containers

> 软件危机的主要原因，把它很不客气地说：在没有机器的时候，编程根本不是问题；当我们有了电脑，编程开始变成问题；而现在我们有巨大的电脑，编程就成为了一个同样巨大的问题。
>
> —艾兹赫尔·戴克斯特拉，谦逊的程序员，《Communications of the ACM》[8]

Goal

Beat Gianter Problem in Programming!

Three Design Patterns

Three Design Patterns

1. Single-container management patterns
2. Single-node, multi-container application patterns
    - Sidecar pattern
    - Ambasssdor pattern
    - Adapter pattern
3. Multi-node application patterns
    - Leader election pattern
    - Work queue pattern
    - Scatter/gather pattern

Single-container management patterns

– 基于容器管理的单容器模式

- Traditional
  - Run()
  - Pause()
  - Stop()
- More Options
  - Status
  - Threads, Stack
  - Lifecycle
  - ...

Single-node, multi-container application patterns

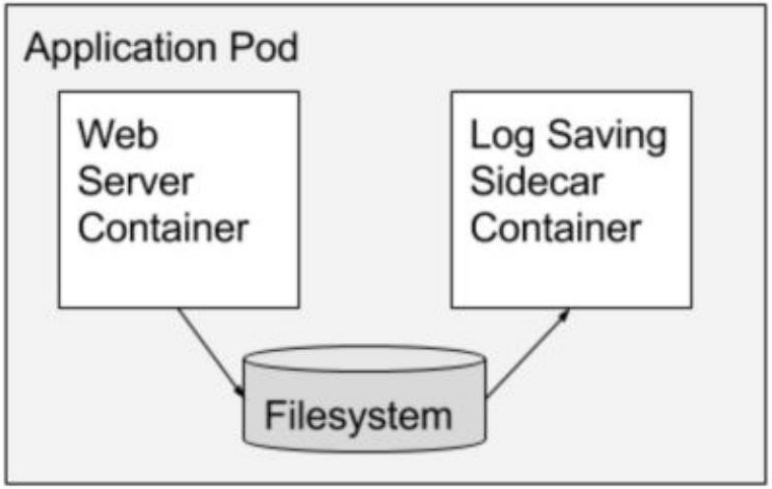
– 用于紧密协作的容器的单节点模式

- Single-node
- Multi-container

Sidecar pattern

– 随航模式

- Main container
  - Web Server
- Sidecar container
  - Log Saving



- 主容器从本地磁盘收集Web服务器日志并将其传输到集群的存储系统

Several Advantages

1.容器是资源计数和分配的单位。

2.容器是packing的单元

- 因此可以将服务和日志存储分隔到不同的容器中，可以很容易地在两个独立的编程队伍之间分配其开发的责任，并且可以进行单独和一起测试

3.容器是重用的单元

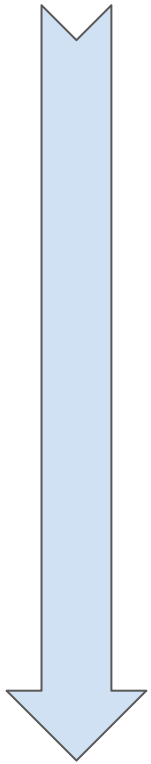
- 所以Sidecars容器可以与不同的主容器进行配对，保证了可重用性

4.容器提供了一个故障遏制的边界

- 这样出现问题时可以使得整个系统进行优雅地降级

5.容器是部署的单位

- 它允许每个功能都可以独立地进行升级或者回滚

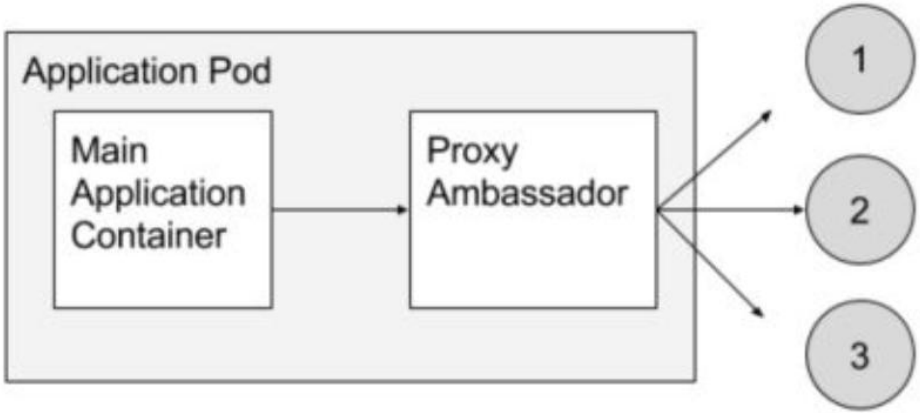


-All mentioned above fit all patterns mentioned in this paper

Ambassador pattern

– 大使模式

- Program to link to localhost
- Test App Indenpendence
- Reuse Ambassador container

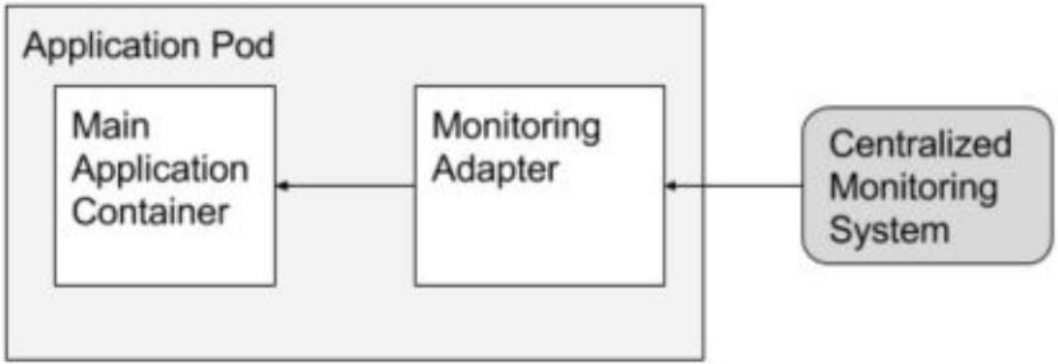


- 应用容器办理业务时只需要跟本Pod的大使打交道就可以了

Adapter pattern

– 适配器模式

- Simplified view/interface
- Consistent interface



- 如果所有应用程序都提供了一致的监视接口，单个监视工具就可以更容易地收集、汇总和显示异构应用程序集中的监控数据

Multi-node application patterns

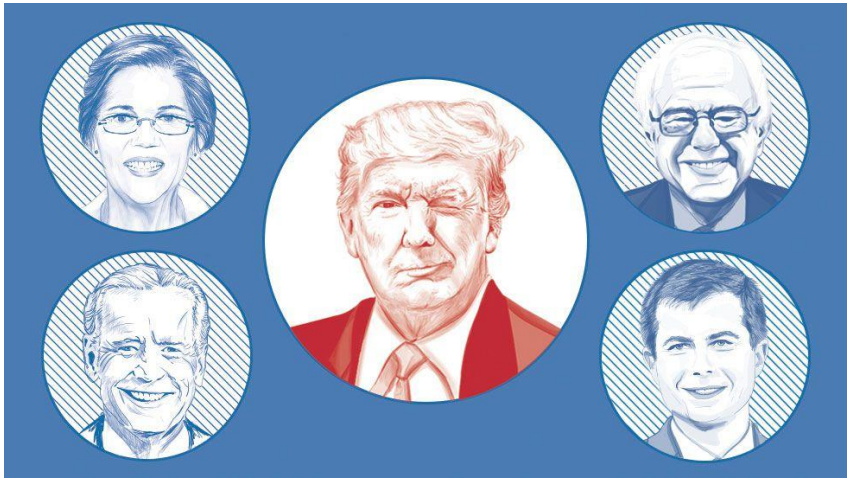
– 用于分布式算法的多节点模式

- Multi-node
- Multi-container

Leader election pattern

– 领导选举模式

- Leader election containers group
  - ▲ based on kinds of languages
- Distributed system applications
  - ▲ based on different languages

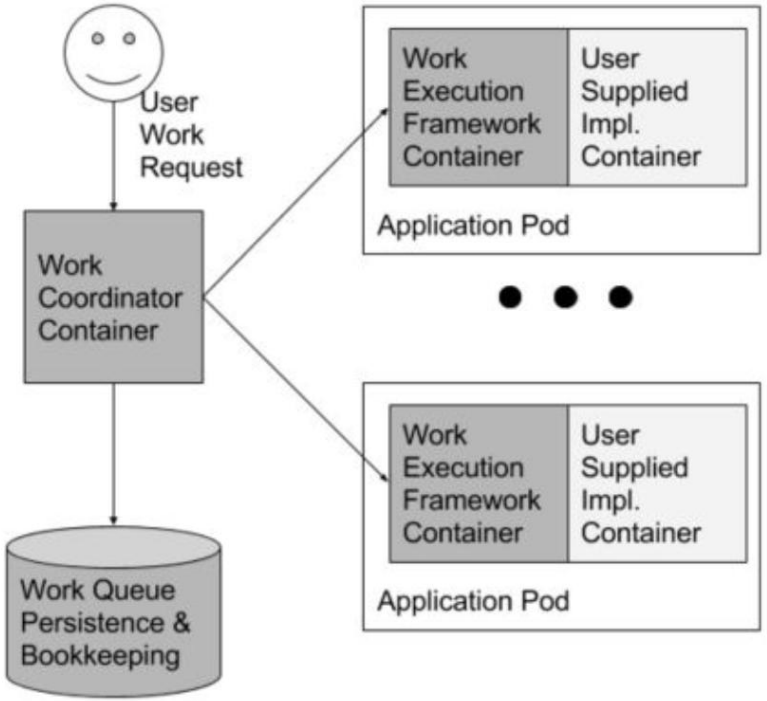


-这些领导选举容器可以构建一次，然后由开发人员重新使用简化的接口

Work queue pattern

– 工作队列模式

- Work Coodinator Container
  - Pod
    - Work Execution Framework
    - User Supplied Implement
  - ...
  - Pod
- Work Queue Persistence & Bookkeeping

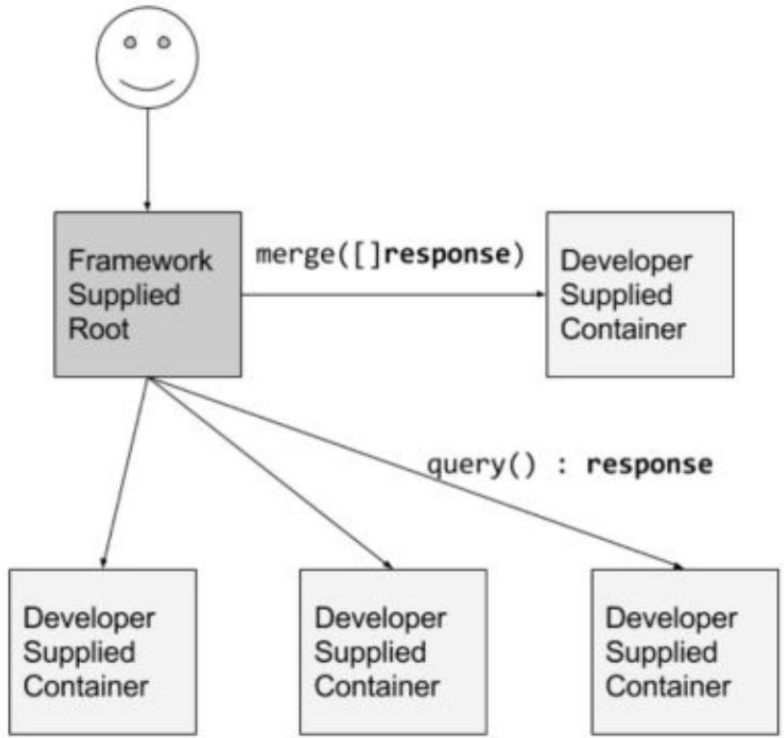


- 所有涉及开发完整工作队列的其他工作都可以由通用工作队列框架来处理，该框架可以在需要此类系统时重用

Scatter/gather pattern

– 分散/聚集模式

- Root Container
  - Distribute requests
  - Collect requests
  - Interact with client
- Node Containers
  - Compute & return results
- Merge Container
  - Merge results, group responses



-通过提供实现这些相对简单的接口的容器，可以很容易地看到用户如何实现任意深度的分散/聚集系统

Related Work

Related Work

- 面向服务的体系结构 SOA
- 微服务架构 Microservices
- 无服务器架构 Serverless

Conclusion & Storming

Conclusion & Storming

- Much as OOP, advantages are supplied
  - Divide tasks
  - Reuse components
- Independent upgrade or degrade
  - Code by different languages
- More standardized & normalized
  - Main theme of industrialisation

- It will be more and more popular!!!

References

References

- [1] 用大白话聊聊分布式系统
- [2] 大家都在说的分布式系统到底是什么？
- [3] 十分钟明白什么是容器技术
- [4] 软件危机
- [5] Notes of <Design patterns for container-based distributed systems>
- [6] 更多的云设计模式
- [7] Designing Distributed Systems

Questions?