CLOUD CONNECT China

IaaS软件设计

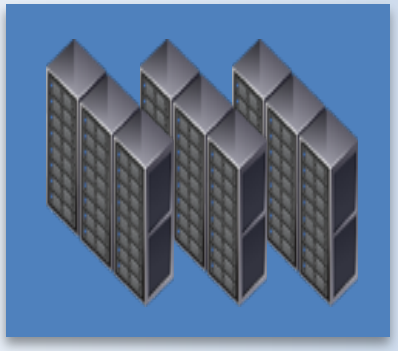
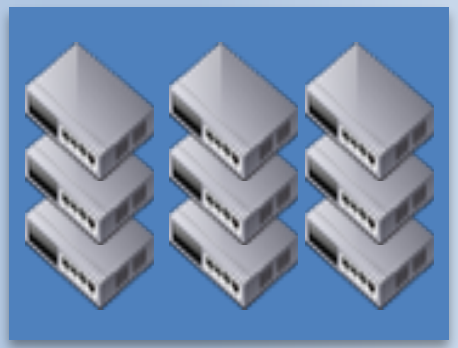
难点与解决方案

张鑫 – ZStack创始人

议题

- 业务范围
- 设计难点
- 当前方案的缺点
- 更优的解决方案

业务范围

附件	Identity	Monitoring	API adapters
	Prediction	Auto Scaling	Alarms
核心	 Compute	 Storage	 Network
子系统	KVM	NAS	Traditional Network
	Xen	SAN	SDN
	VMWare	SDS	NFV

业务范围

公有云IaaS软件 公有云厂商的IaaS软件是为自己的数据中心设计	通用IaaS软件 通用IaaS软件是为所有客户的数据中心设计
••封闭的业务范围 ••无产品化的要求和意愿 ••强大的运维团队 ••专有设计，为某些方面做特殊优化，例如性能、可伸缩性	••开放的业务范围 ••必须产品化 ••客户水平参差不齐 ••通用设计，必须满足不可预知的客户需求

公有云IaaS软件围绕自己的业务做定制设计，目的是服务自己业务的打造一个最优系统。

通用IaaS软件为了开放业务范围设计，要能满足未知的客户需求，通常只能打造成次优系统。

业务范围

大部分*IaaS*软件取名为*x-stack*，因为他们的本质是集成 软件，由不同的子系统堆叠而成。

CLOUD CONNECT China

设计难点

客户的痛点往往也是设计的难点

设计难点

- 管理数十万物理机
- 管理千万级的虚拟机
- 服务数万并发API

IaaS系统里任务的执行路径很长，导致系统整体并发度低。

案例1: VDI厂商需要IaaS软件能够在数分钟内启动数千虚拟机。

设计难点

- 当错误发生时，要能回滚已完成操作，保证整个系统的元数据一致性
- 内核要能够在快速的功能迭代中保持稳定，避免频繁产生退化的bug
- 稳定性不受IaaS规模的影响

软件不稳定的根源在于代码在开发过程中不断更改，缺乏一个稳定的code base

案例 2: 很多IaaS用户惧怕升级，因为新版本往往会带来很多bug，导致生产环境不稳定

设计难点

- 当管理大量物理设备时，如何安装agent、安装依赖、配置系统是一大难题
- 全自动化交付、全API交付，无手动操作
- 无缝、稳定的升级

客户希望软件能够简单到下载 - 安装 - 运行。

案例3: 很多客户因为IaaS软件的部署、升级的复杂性而放弃了他们的云化计划。

设计难点

- 子系统通常都有厂商特有的功能，但用户需要的是高度抽象化的功能
- 通用IaaS软件通常会遇到非常多未知的需求
- 上层软件(PaaS/SaaS)、第三方软件希望能够与IaaS软件无缝集成

灵活性是产品在激烈竞争中胜出的一个关键。

案例4: 已经有IaaS先驱由于缺乏灵活性，无法拥抱传统的IT企业需求而成了前浪，倒在了沙滩上。

设计难点

灵活性
VS
易用性

可伸缩性
VS
易用性

灵活性
VS
稳定性

崩溃! 这些设计目标自相矛盾!

设计难点

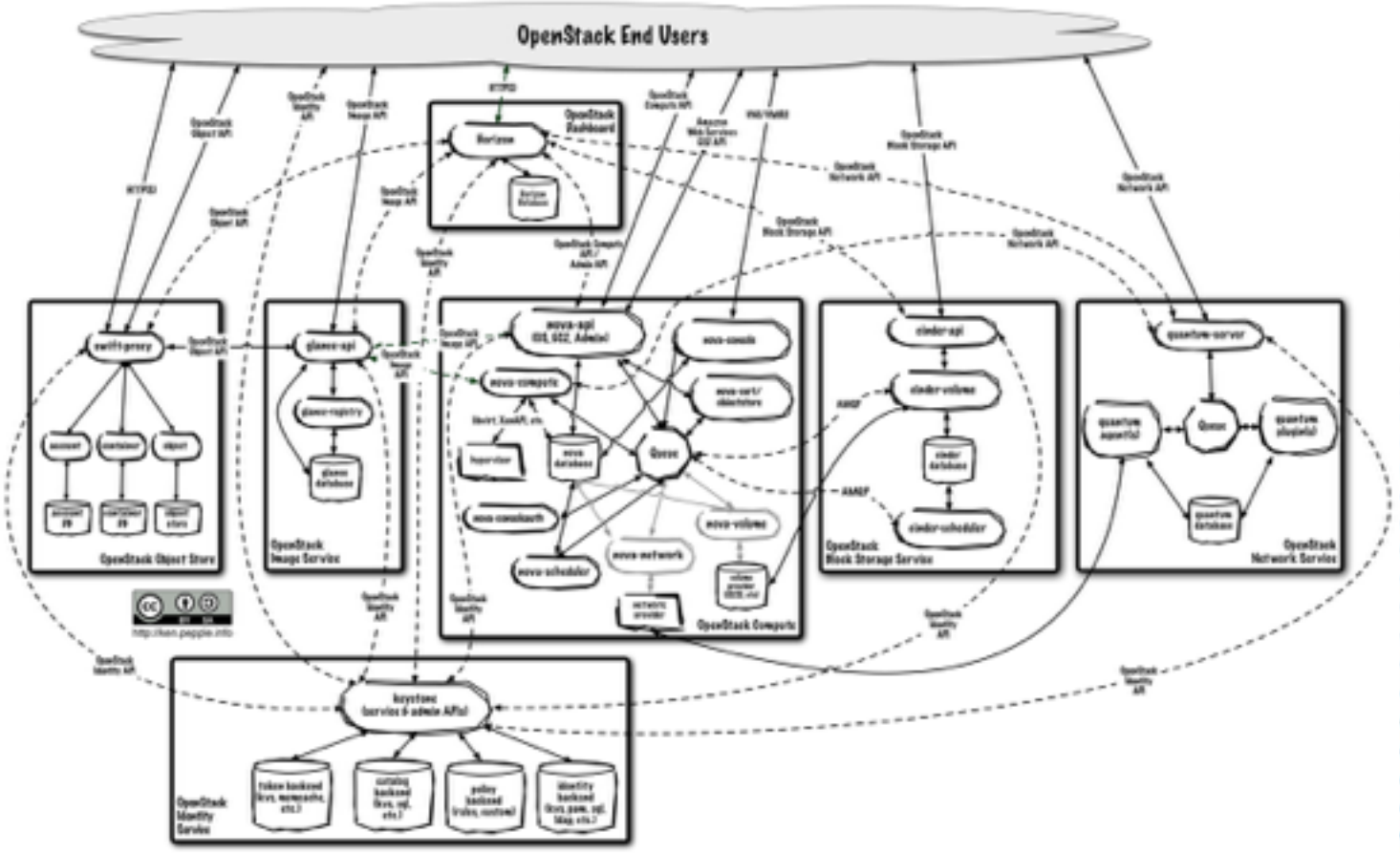
产品化，产品化，产品化！

CLOUD CONNECT China

现有方案

集中式架构赢了易用性，却失去了灵活性、稳定性、可伸缩性。

现有方案



纯微服务架构有一定的灵活性，但易用性、稳定性、可伸缩性不足。

现有方案

世上没有万能的银子弹。一个折中的混合架构才是解决之道。

更好的方案- 易用性

安装

下载即可部署安装

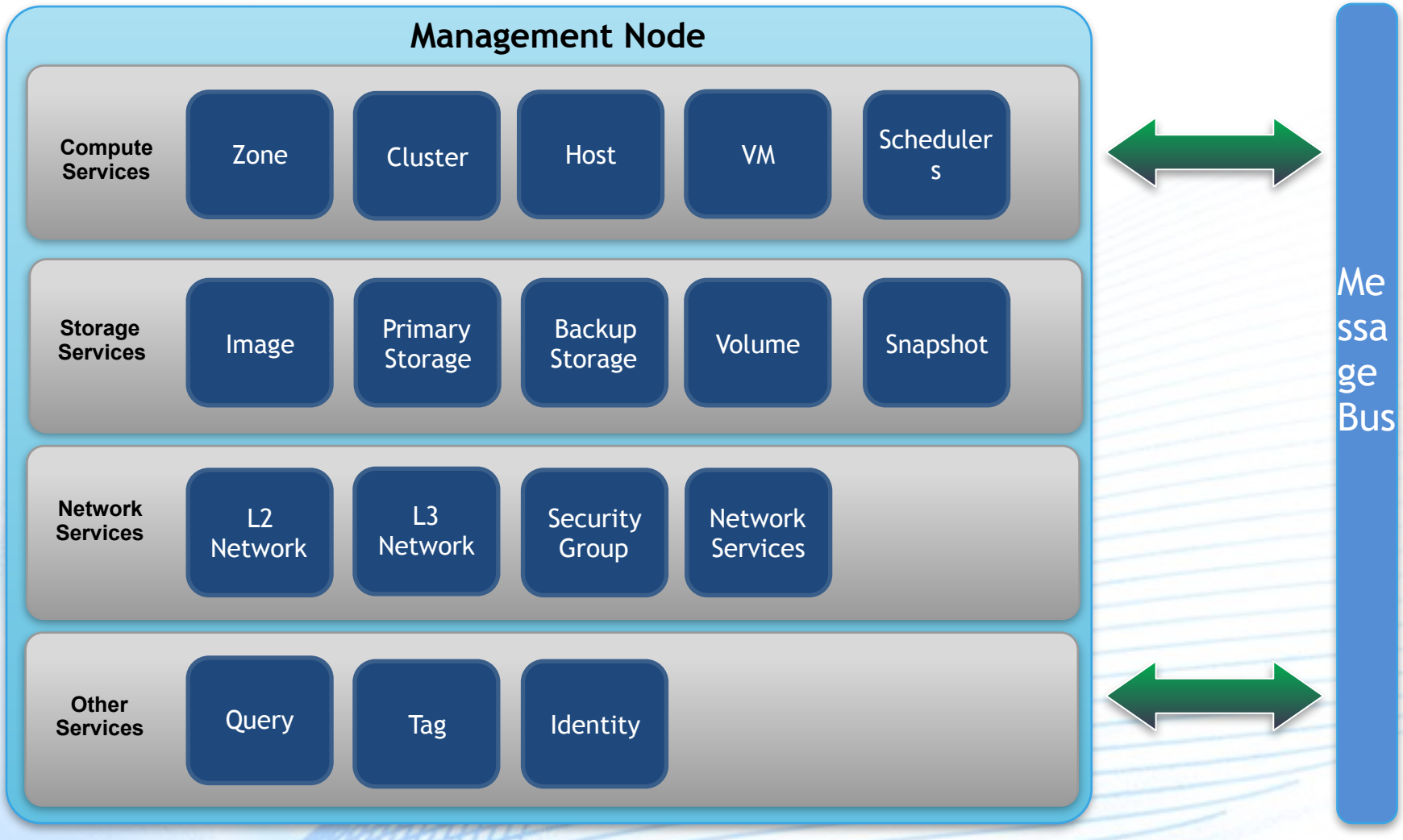
部署

全自动部署
无需手动配置

升级

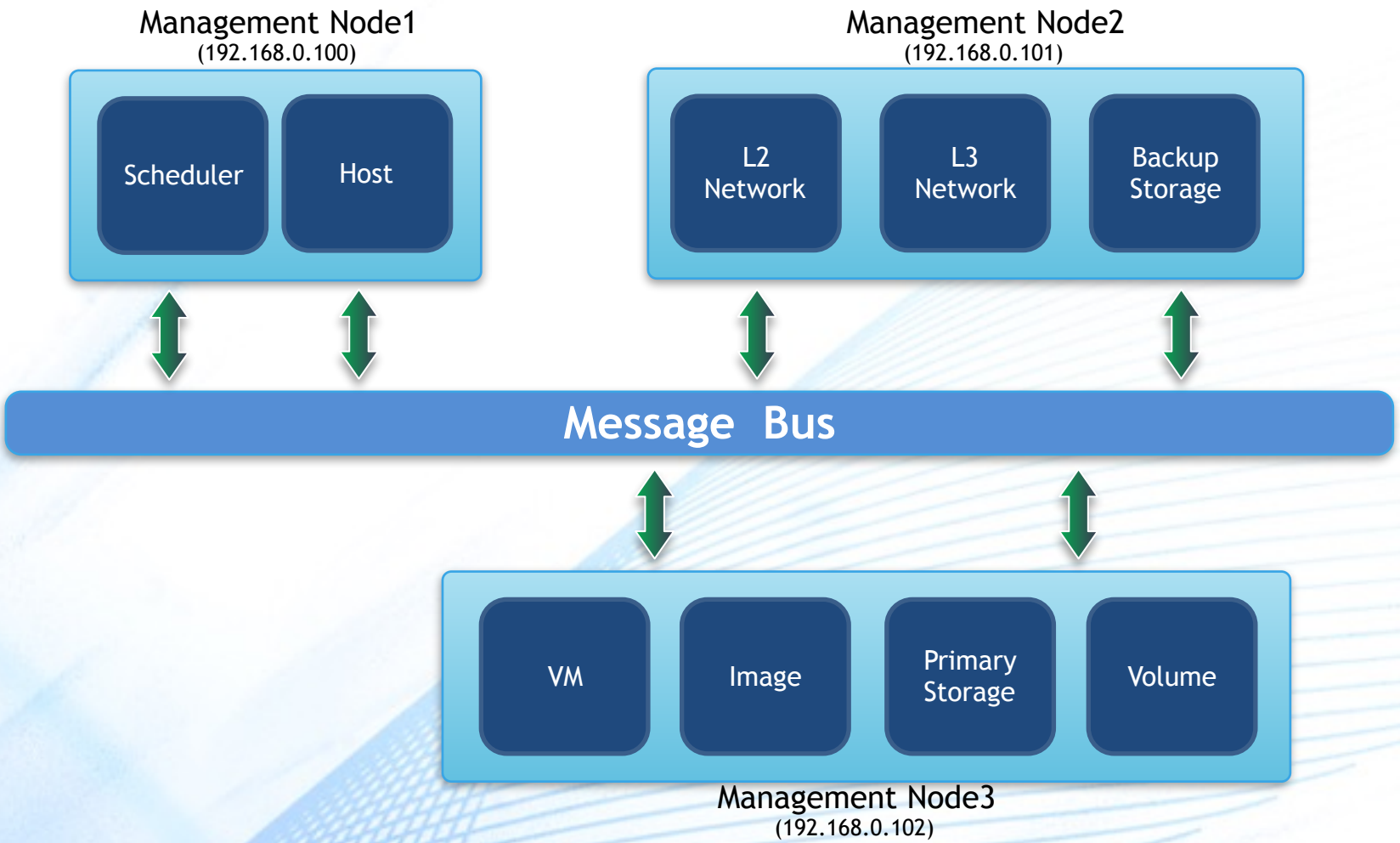
无缝升级
无需手动一个个的升级组件

更好的方案- 易用性



进程内微服务架构

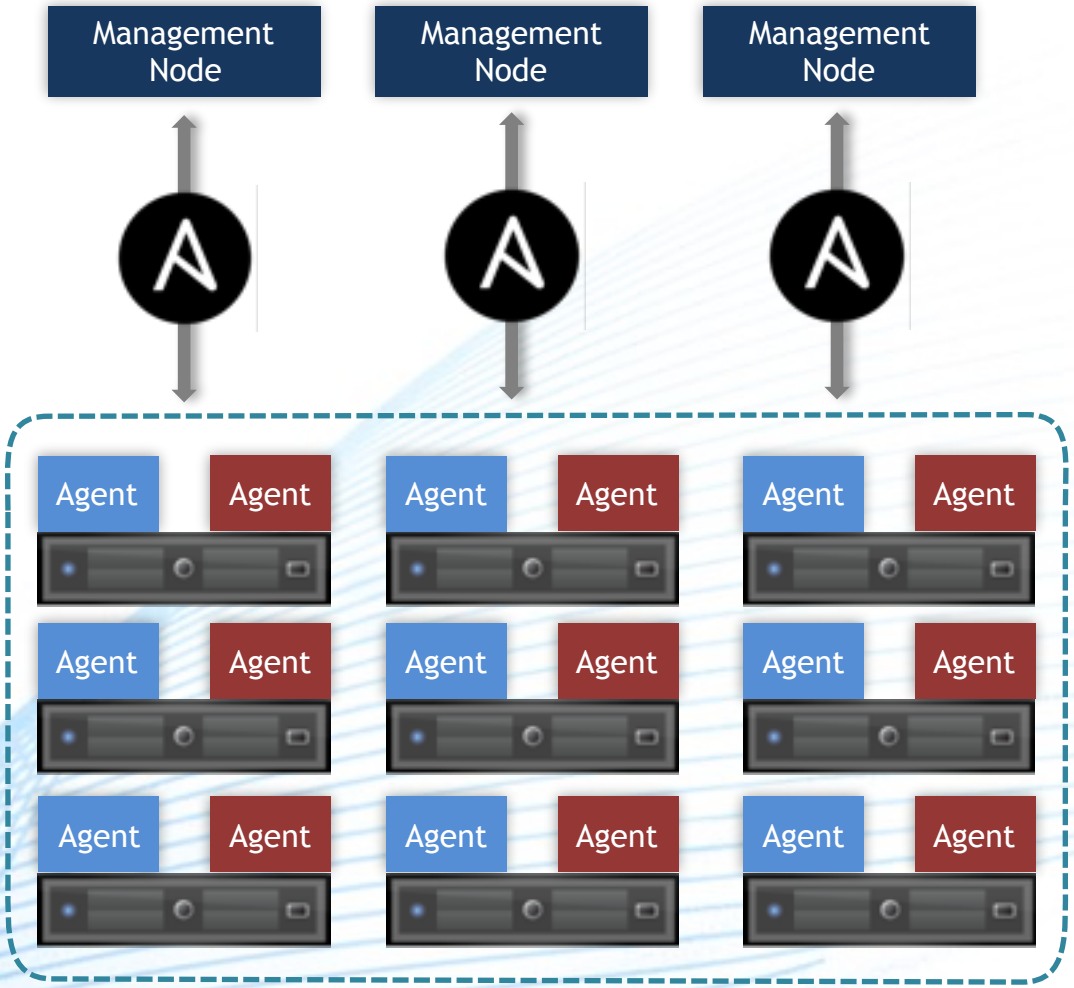
更好的方案- 易用性



运行在不同机上的管理节点内的微服务相互之间通过消息总线通讯。

更好的方案- 易用性

- 管理节点调用Ansible去安装依赖、配置物理设备、部署agent
- 过程对用户透明，无需用户干预
- 可 通过重连agent自动对agent进行升级



无缝集成Ansible

更好的方案- 易用性

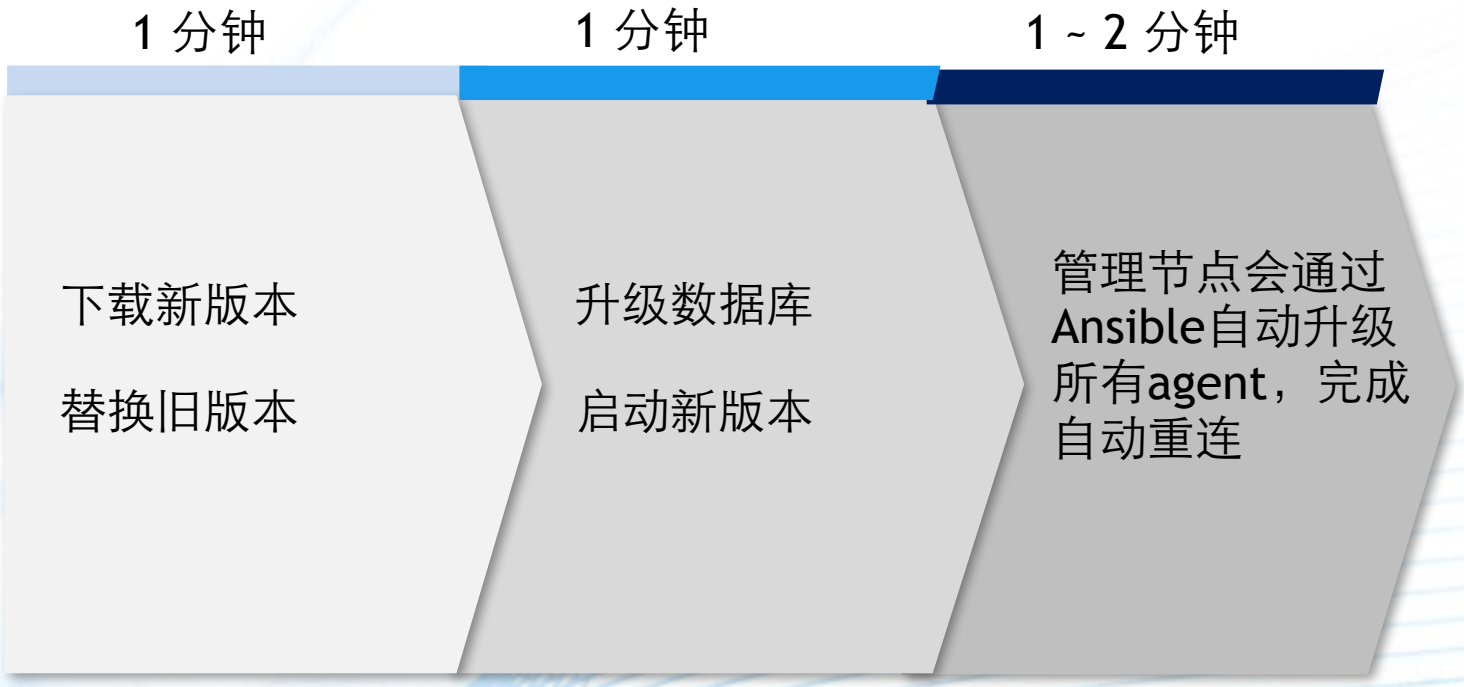
```
- include: zstacklib.yaml

- name: create root directories
  shell: "mkdir -p {{item}}"
  with_items:
    - "{{kvm_root}}"
    - "{{virtenv_path}}"

- name: install kvm related packages on RedHat based OS
  when: ansible_os_family == 'RedHat'
  yum: name="{{item}}"
  with_items:
    - qemu-kvm
    - bridge-utils
    - wget
    - qemu-img
    - libvirt-python
    - libvirt
    - nfs-utils
    - vconfig
    - libvirt-client
    - net-tools
    - iscsi-initiator-utils

- name: install kvm related packages on Debian based OS
  when: ansible_os_family == 'Debian'
  apt: pkg="{{item}}"
  with_items:
```

更好的方案- 易用性



三条命令完成升级:

1. zstack-ctl upgrade_management_node
2. zstack-ctl upgrade_db
3. zstack-ctl start_node

更好的方案- 可伸缩性

物理机

数十万台物理机

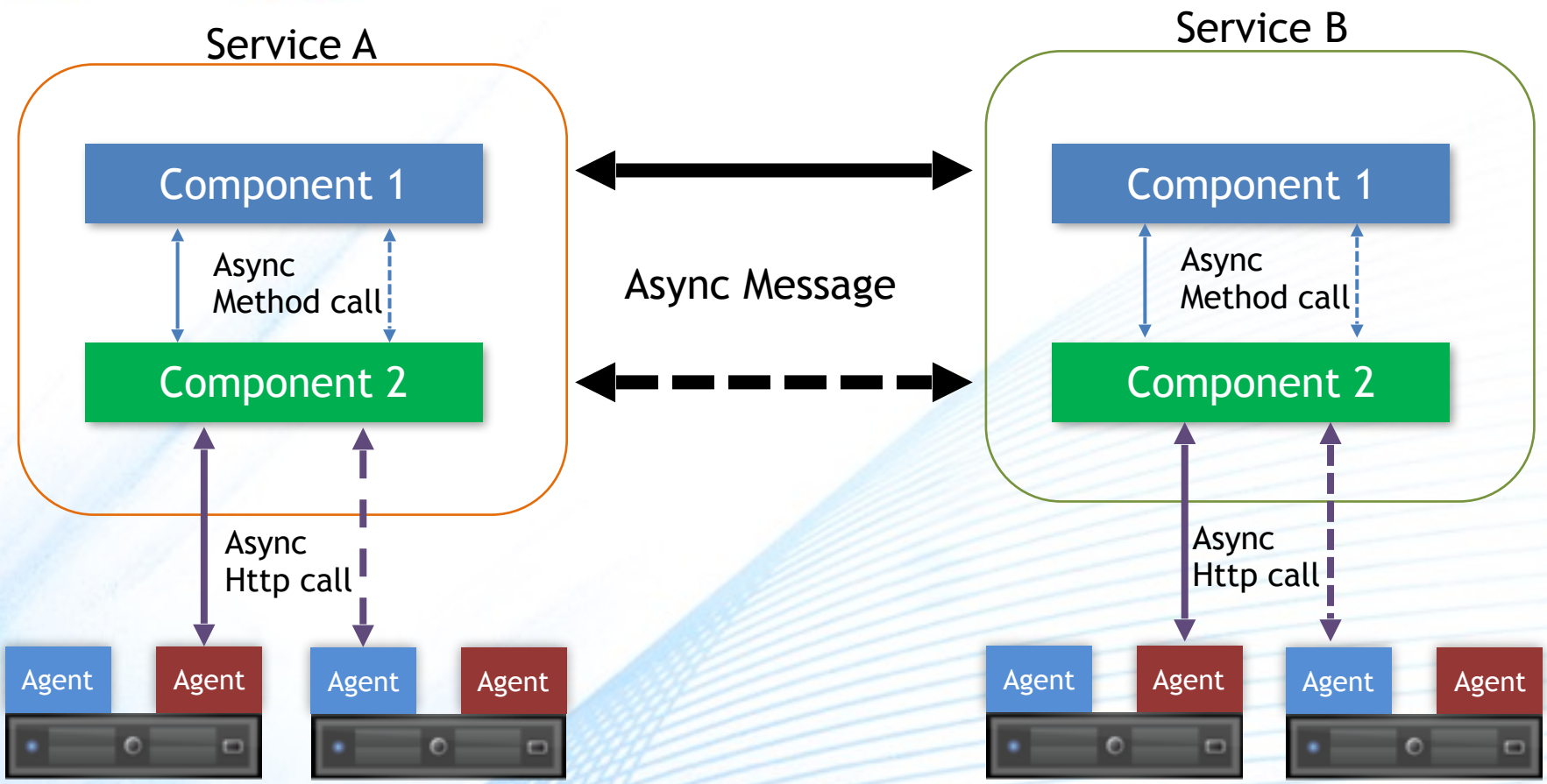
虚拟机

百万级虚拟机

并发APIs

数万并发API

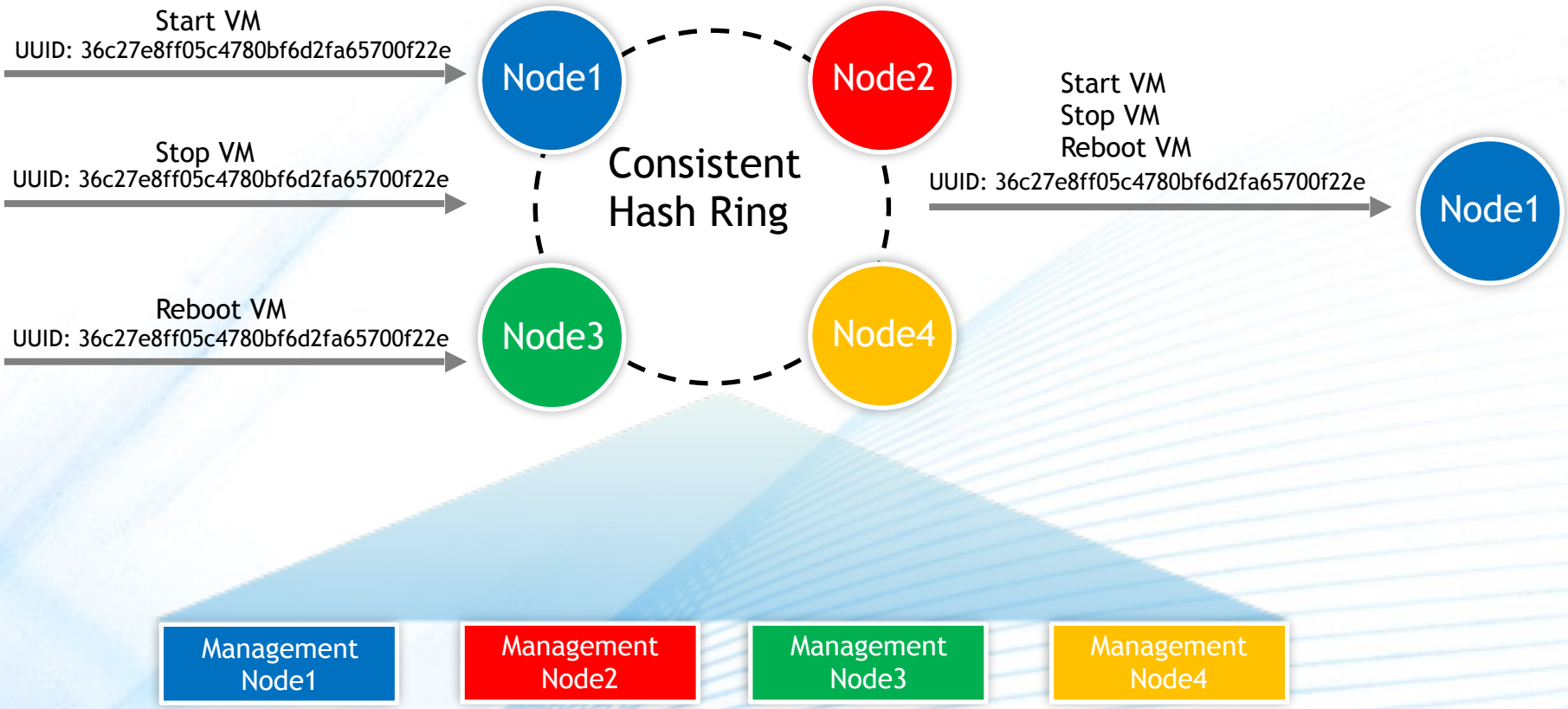
更好的方案- 可伸缩性



全异步架构

500个线程的线程池可以服务10000+并发API。

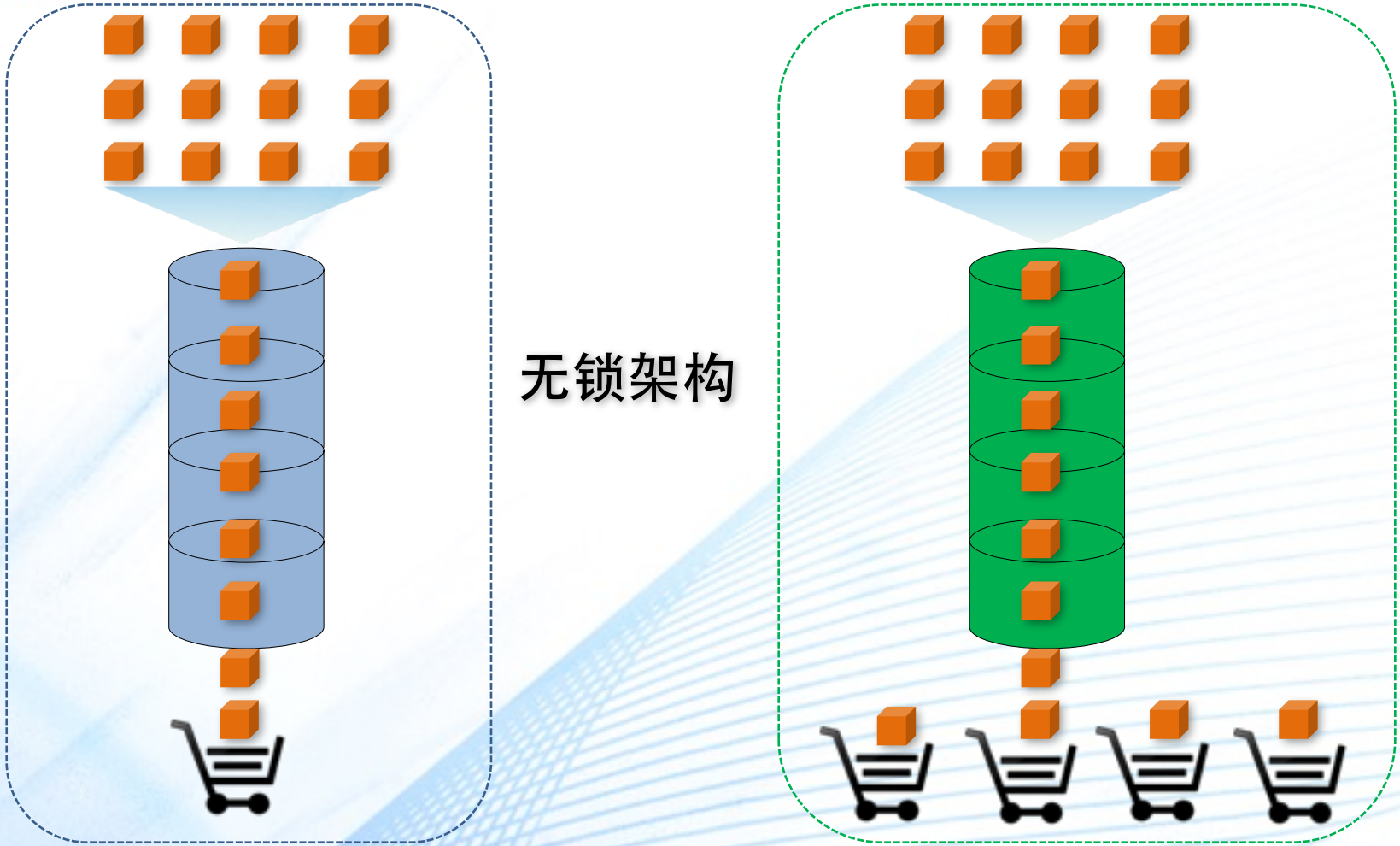
更好的方案- 可伸缩性



无状态服务架构

一致性哈希环保证对于同一个资源的请求只会被同一个微服务处理。

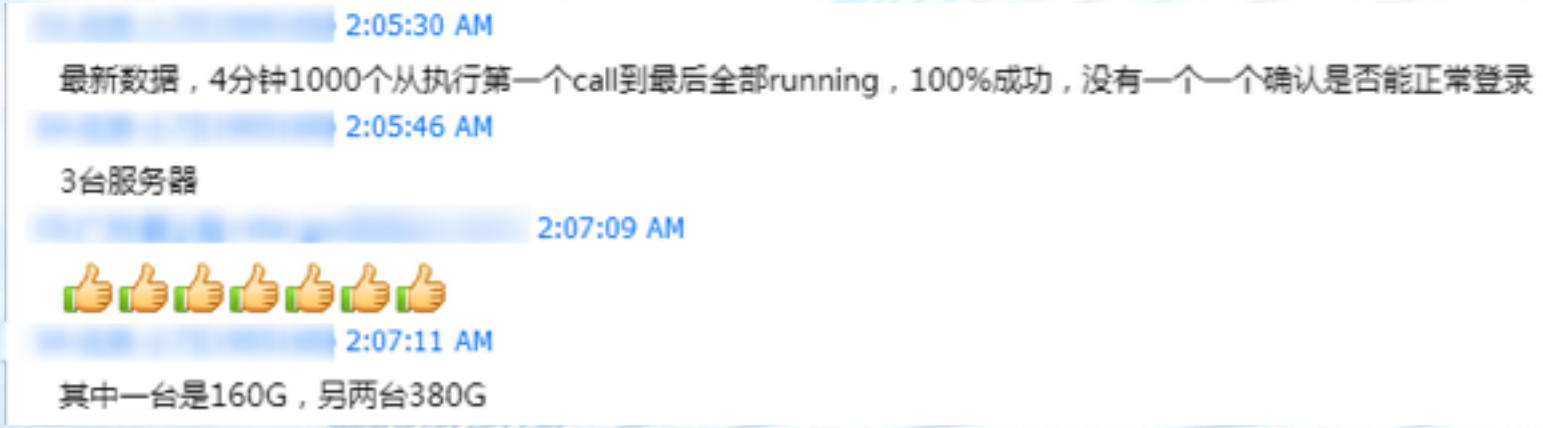
更好的方案- 可伸缩性



每一个资源的每一种操作都可以通过队列来控制并发度。

更好的方案 - 可伸缩性

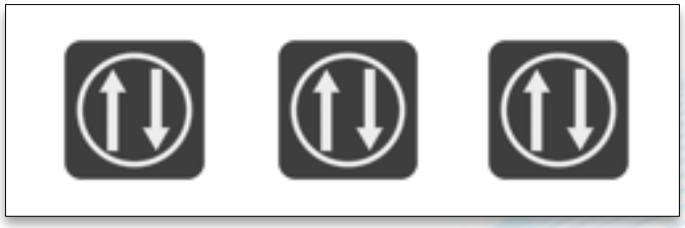
测试手段	结果
模拟器	500并发API在2.6秒内完成创建500个虚拟机
真实环境	1000个并发API在4分钟内完成创建1000个虚拟机



更好的方案- 可伸缩性

QueryVmInstance
vmNics.ip=192.168.0.249

Query API



Query Service

Translate to: select vm from VmInstanceVO vm where
…

类SQL的查询API

超过4百万个单项查询条件；组合查询条件为4百万的阶乘。

更好的方案- 可伸缩性

```
>>>QueryVmInstance vmNics.
[Query Conditions:]
vmNics.eip.             vmNics.l3Network.       vmNics.portForwarding.  vmNics.securityGroup.   vmNics.vmInstance.

vmNics.__systemTag__=   vmNics.__userTag__=     vmNics.createDate=      vmNics.deviceId=        vmNics.gateway=         vmNics.ip=
vmNics.lastOpDate=      vmNics.mac=             vmNics.metaData=        vmNics.netmask=         vmNics.uuid=            vmNics.vmInstan


>>>QueryVmInstance vmNics.ip=192.168.0.249
2015-04-08 09:21:12,817 DEBUG [apibinding.api] async call[url: http://localhost:8080/zstack/api/, request: {"org.zstack.header.vm.APIQu
acdbe93fa044b31578d"}, "conditions": [{"name": "vmNics.ip", "value": "192.168.0.249", "op": "="}]}}]
{
    "inventories": [
        {
            "allVolumes": [
                {
                    "createDate": "Apr 6, 2015 9:40:28 AM",
                    "description": "Root volume for VM[uuid:ffe2bd05dd4347459e2fd914fc72a1fa]",
                    "deviceId": 0,
                    "format": "qcow2",
                    "installPath": "/opt/zstack/nfsprimarystorage/prim-e513c60d224640a88888366ba836e01e/rootVolumes/acct-36c27e8ff05c47
5/22955a2cea85423893e088806c5b4d65.qcow2",
                    "lastOpDate": "Apr 6, 2015 9:40:28 AM",
                    "name": "ROOT-for-virtualRouter.l3.bda24d47959a47bfa9383b7bee769452",
                    "primaryStorageUuid": "e513c60d224640a88888366ba836e01e",
                    "rootImageUuid": "132e66bb729b44659a33c5332a0c20f4",
                    "size": 445579264,
                    "state": "Enabled",
                    "status": "Ready",
                    "type": "Root",
                    "uuid": "22955a2cea85423893e088806c5b4d65",
                    "vmInstanceUuid": "ffe2bd05dd4347459e2fd914fc72a1fa"
                }
```

更好的方案- 灵活性

稳定的内核

添加新功能不应该改动已有代码

稳定的数据库

新功能应该不改变现有数据库或改动很小

第三方集成

第三方软件应该能够无缝集成

共性与特性

API既能够抽象共性，也能反应厂商产品特性

更好的方案- 灵活性

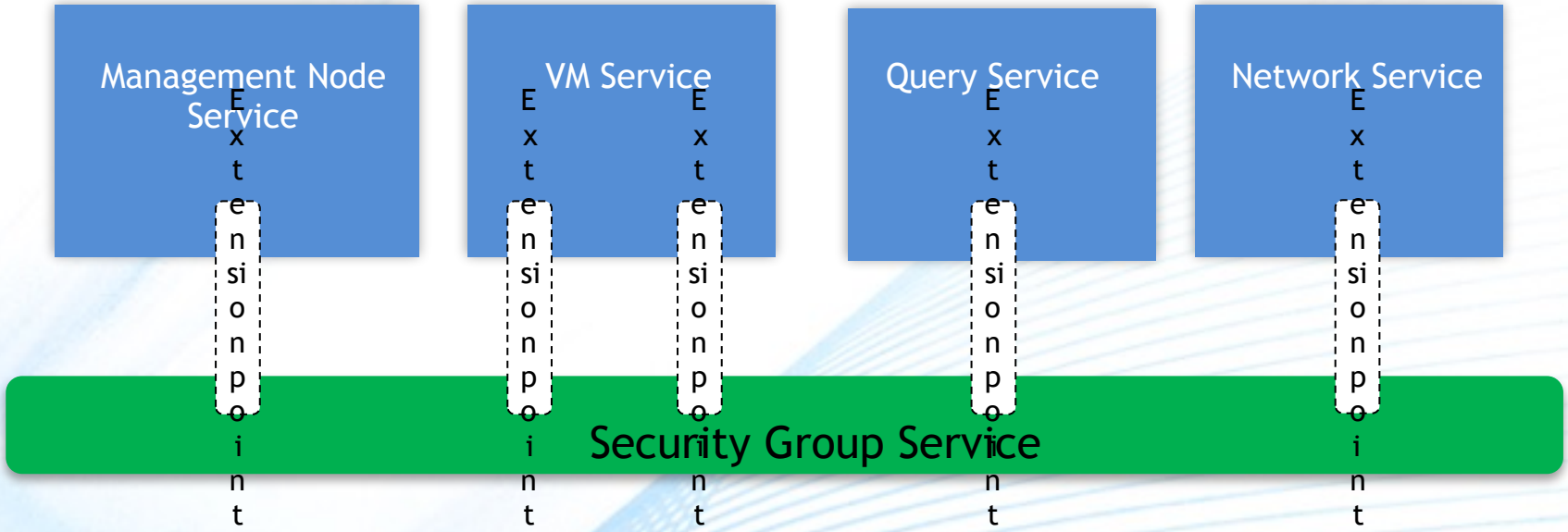
策略型插件

全插件架构

策略型插件多为子系统驱动，编译成单独的JAR文件。

更好的方案- 灵活性

观察者插件



全插件架构

观察者插件通过扩展点挂入已有组件，扩展功能。编译成单独的JAR文件。

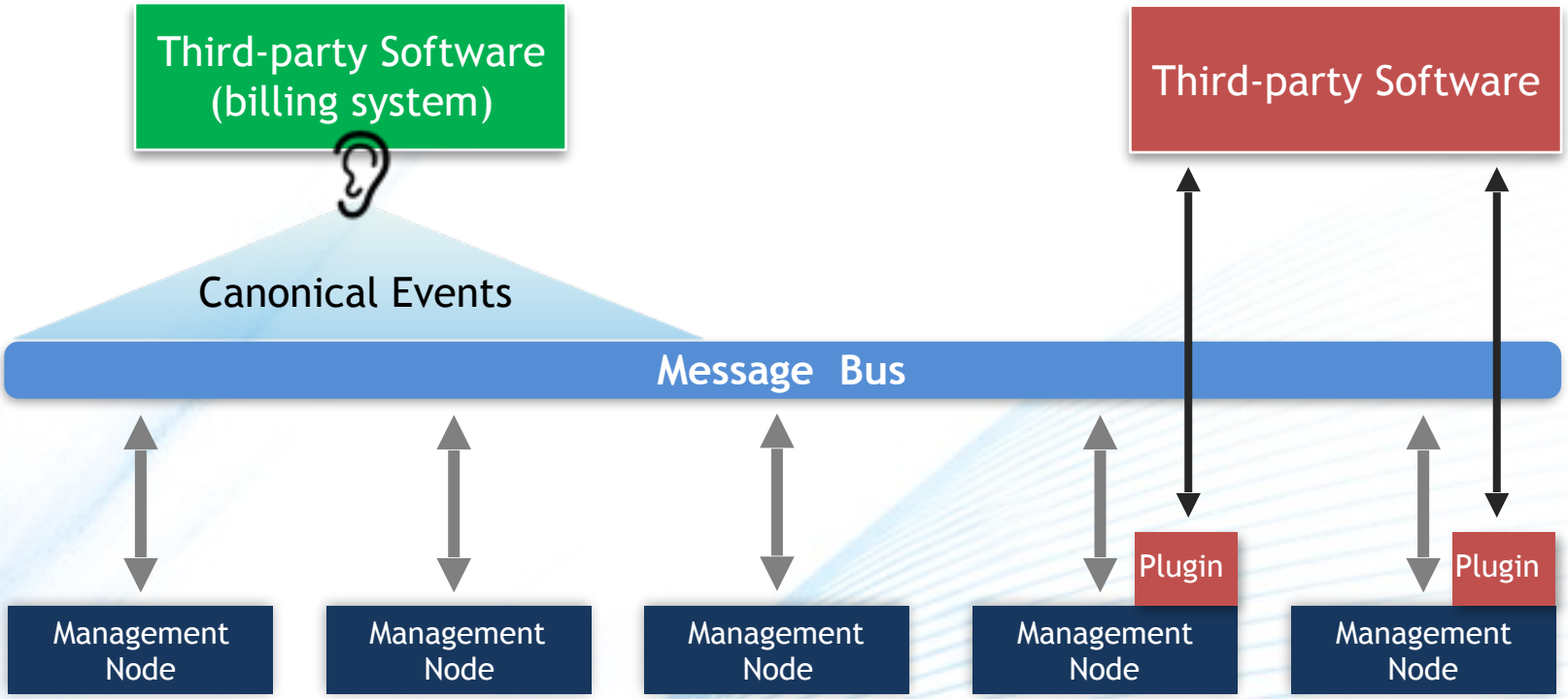
CLOUD CONNECT China

更好的方案- 灵活性

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx" xmlns:zstack="http://zstack.org/schema/zstack"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
        http://zstack.org/schema/zstack
        http://zstack.org/schema/zstack/plugin.xsd"
    default-init-method="init" default-destroy-method="destroy">

    <bean id="SecurityGroupManager"
        class="org.zstack.network.securitygroup.SecurityGroupManagerImpl">
        <zstack:plugin>
            <zstack:extension interface="org.zstack.header.Component" />
            <zstack:extension interface="org.zstack.header.Service" />
            <zstack:extension interface="org.zstack.header.managementnode.ManagementNodeChangeListener" />
            <zstack:extension interface="org.zstack.header.vm.VmInstanceMigrateExtensionPoint" />
            <zstack:extension interface="org.zstack.header.query.AddExpandedQueryExtensionPoint" />
            <zstack:extension interface="org.zstack.header.identity.ReportQuotaExtensionPoint" />
        </zstack:plugin>
    </bean>
```

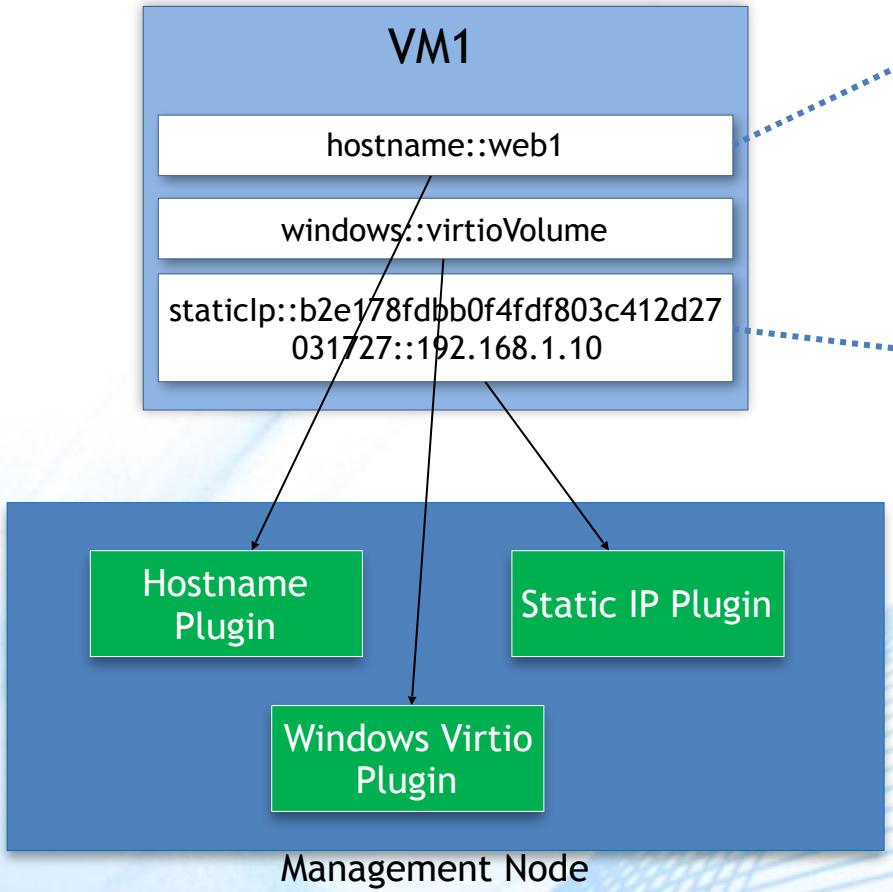
更好的方案- 灵活性



全插件架构

进程外插件可以用任何语言编写，通过消息总线跟管理节点通讯。

更好的方案- 灵活性



Key-value DB Table

Key	Value
b2e178fdbb0f4fdf803c412d27031727	hostname::web1
b2e178fdbb0f4fdf803c412d27031727	windows::virtioVolume
b2e178fdbb0f4fdf803c412d27031727	staticIp::b2e178fdbb0f4fdf803c412d27031727::192.168.1.10

系统标签

```
>>>CreateVmInstance name=vm
systemTags=hostname::web1,windowsVirtioVolume
```

更好的方案 - 稳定性

稳定的code base

添加功能不引起退化性问题

元数据一致性

元数据在错误发生后保证一致性

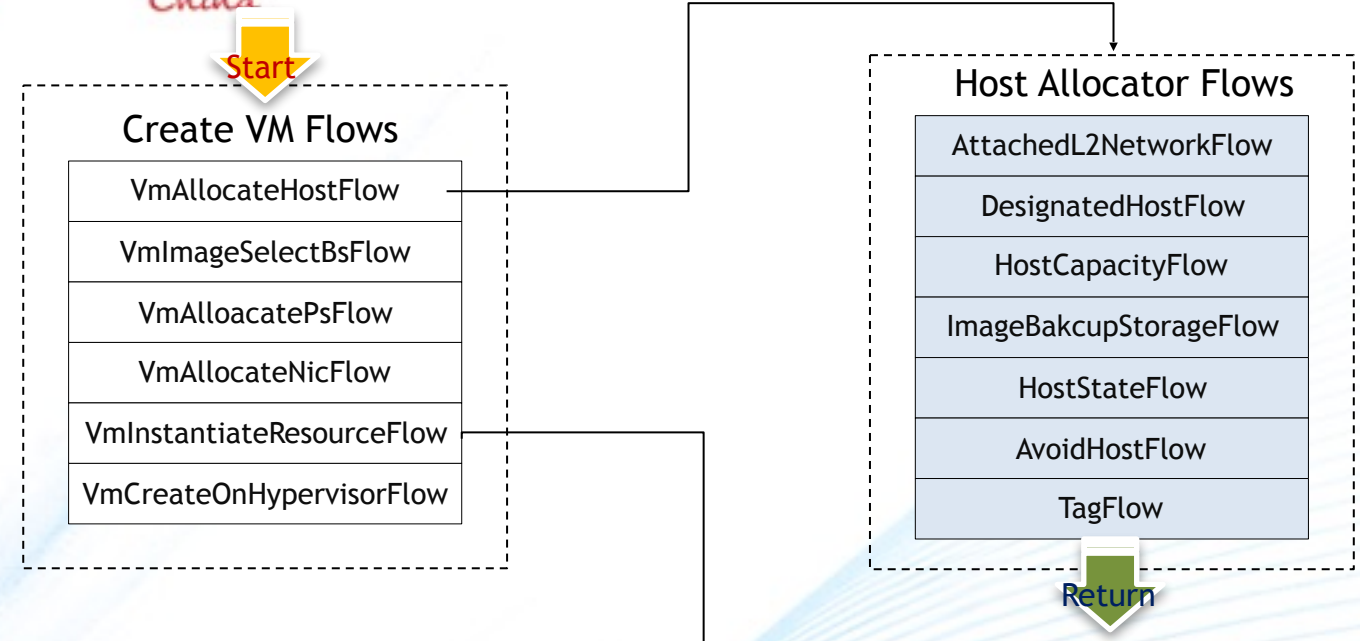
无副作用

操作不遗留副作用在系统

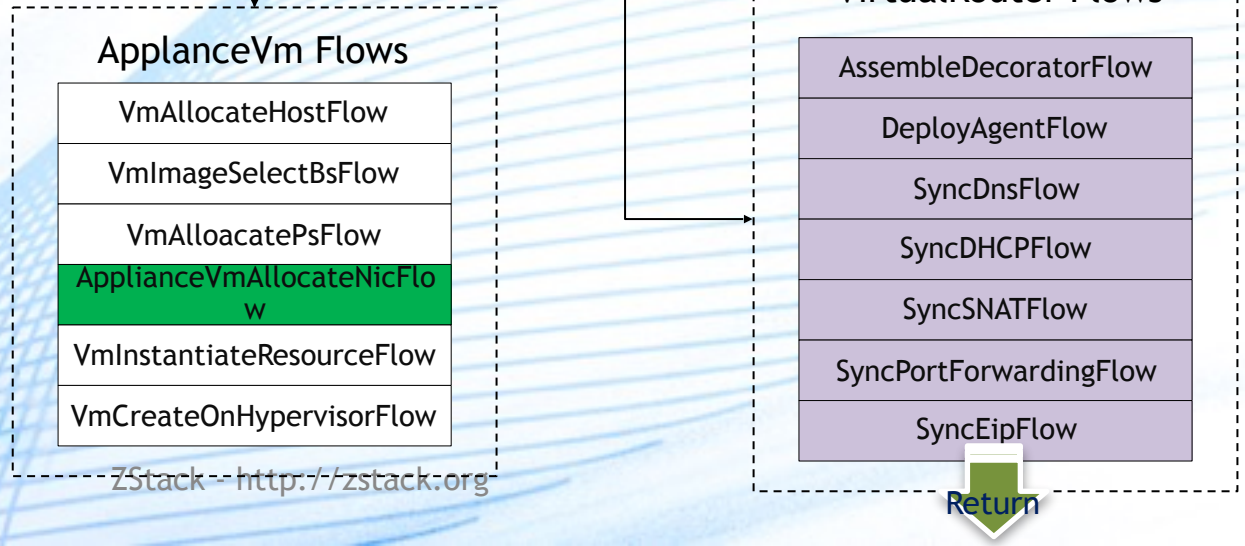
自动化测试

由全自动化测试系统保证质量

更好的方案 - 稳定性



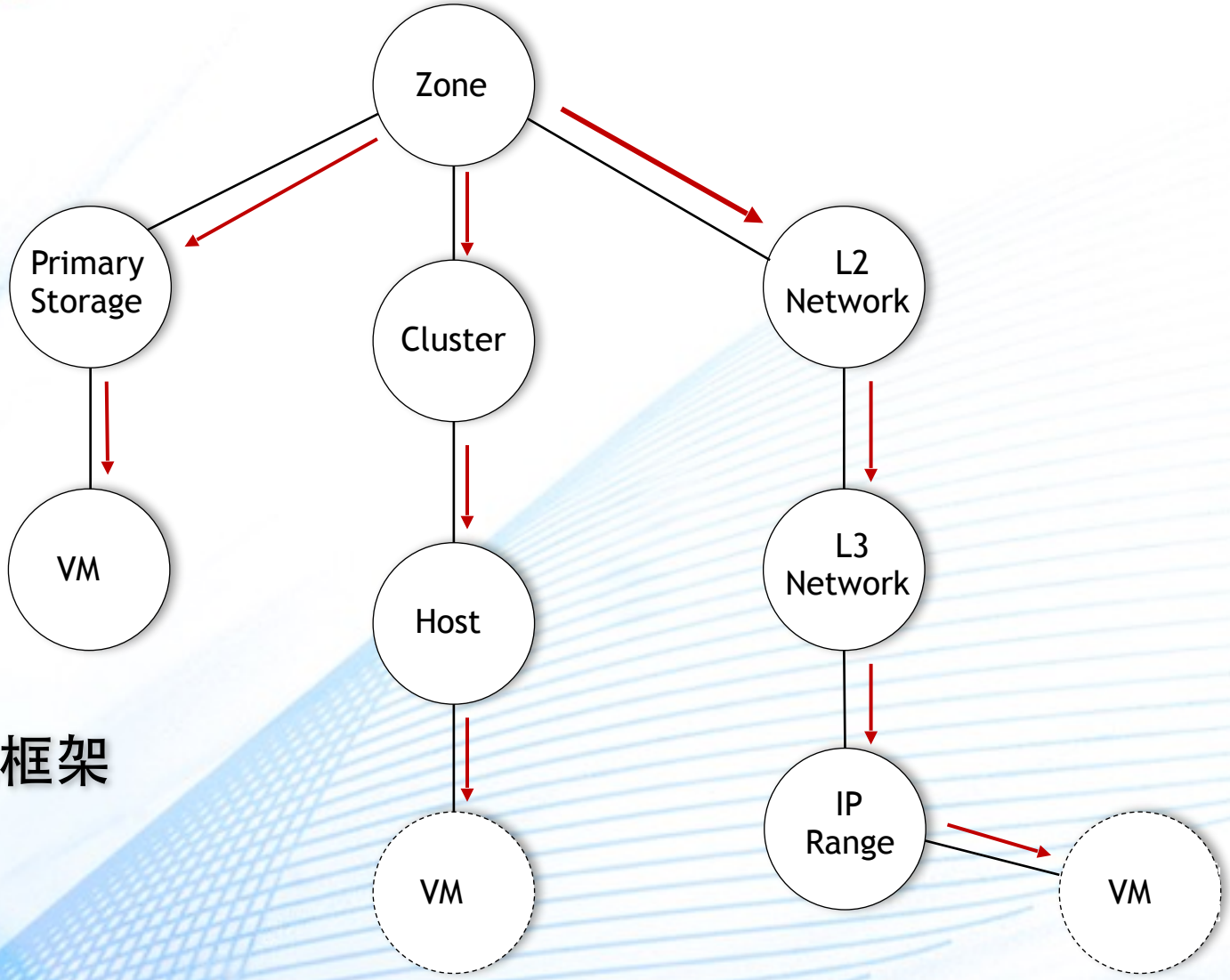
工作流引擎



更好的方案 - 稳定性

```
<bean id="VmInstanceManager" class="org.zstack.compute.vm.VmInstanceManagerImpl">
    <property name="createVmWorkFlowElements">
        <list>
            <value>org.zstack.compute.vm.VmAllocateHostFlow</value>
            <value>org.zstack.compute.vm.VmImageSelectBackupStorageFlow</value>
            <value>org.zstack.compute.vm.VmAllocatePrimaryStorageFlow</value>
            <value>org.zstack.compute.vm.VmAllocateVolumeFlow</value>
            <value>org.zstack.compute.vm.VmAllocateNicFlow</value>
            <value>org.zstack.compute.vm.VmInstantiateResourcePreFlow</value>
            <value>org.zstack.compute.vm.VmCreateOnHypervisorFlow</value>
            <value>org.zstack.compute.vm.VmInstantiateResourcePostFlow</value>
        </list>
    </property>
    <property name="stopVmWorkFlowElements">
        <list>
            <value>org.zstack.compute.vm.VmStopOnHypervisorFlow</value>
            <value>org.zstack.compute.vm.VmReturnHostFlow</value>
            <value>org.zstack.compute.vm.VmReleaseResourceFlow</value>
        </list>
    </property>
    <property name="rebootVmWorkFlowElements">
        <list>
            <value>org.zstack.compute.vm.VmRebootOnHypervisorFlow</value>
        </list>
    </property>
    <property name="startVmWorkFlowElements">
        <list>
            <value>org.zstack.compute.vm.VmAllocateHostForStoppedVmFlow</value>
            <value>org.zstack.compute.vm.VmAllocateNicForStartingVmFlow</value>
            <value>org.zstack.compute.vm.VmInstantiateResourcePreFlow</value>
```

更好的方案 - 稳定性



瀑布式框架

资源可以通过实现插件的方式选择加入这个瀑布框架。

更好的方案 - 稳定性

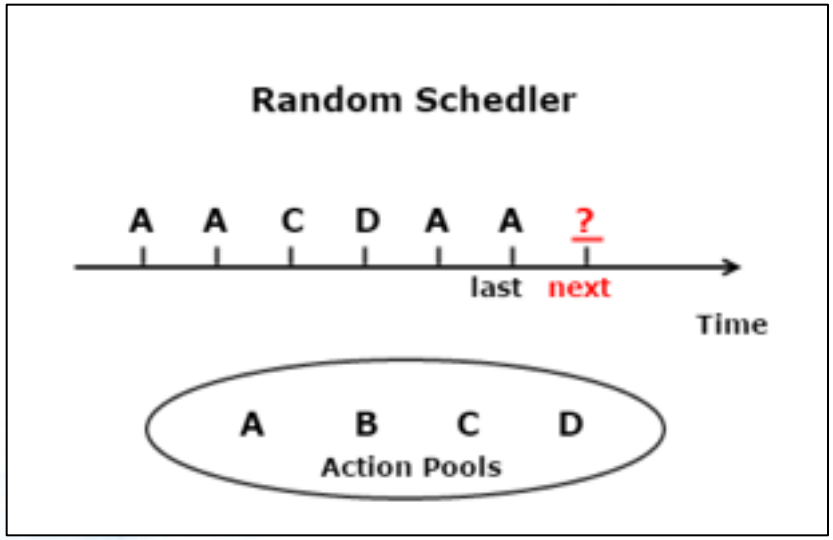
Test System	Details	Period
Integration test	813 cases	~15 hours
System test	984 cases	~24 hours
Model-based test	8 cases	3 ~ 5 days

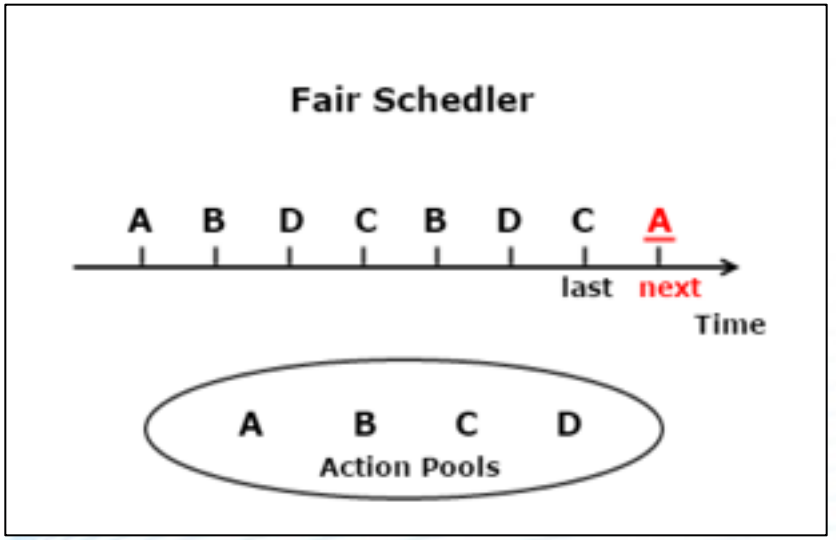
```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running org.zstack.test.UnitTestSuite
2015-08-26 01:10:52,679 INFO  [UnitTestSuite] (main) use configure file: UnitTestSuiteConfig.xml
2015-08-26 01:10:53,066 INFO  [UnitTestSuite] (main) There are total 813 test cases to run
0.TestCloudBusSend ..... [ Success 00:17 ]
1.TestCloudBusSendCallback ..... [ Success 00:17 ]
2.TestCloudBusSendCallbackTimeout ..... [ Success 00:20 ]
3.TestCloudBusSendMultiMsg ..... [ Success 00:17 ]
4.TestCloudBusSendMultiMsg1 ..... [ Success 00:17 ]
5.TestCloudBusSendMultiMsg2 ..... [ Success 00:27 ]
```

全自动化测试

ZStack从最开始就是测试驱动开发，测试新功能的唯一方法是写测试用例。

更好的方案 - 稳定性

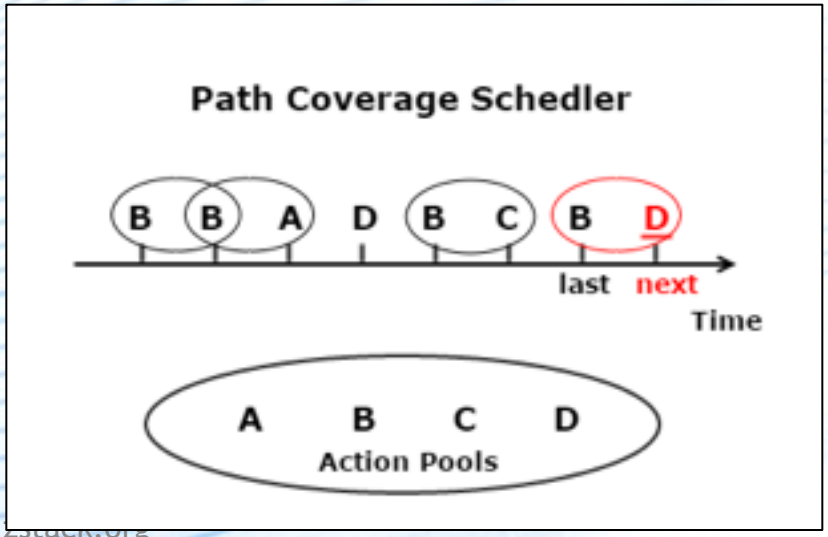




三种用例生成算法

数天的测试时间产生几十G的日志文件

日志重放软件可以自动回放操作，恢复错误发生环境



总结

伽利略：真理总是一目了然，问题是如何发现它。

CLOUD CONNECT China

提问?

谢谢！