中国科学院软件研究所

面向大数据的多源异构云计算平台

概要设计说明书

文档状态	[ ] 初稿 [ ] 评审通过 [ ] 修改 [ ] 发布 [ ] 作废	**作者**	网驰项目组
		部门	
		完成日期	

文档信息：

文档名称	架构设计说明书	**文档编号**	
文档存放			
发行者	中国科学院软件研究所	**发行日期**	2016-6-1

文档变更记录：

修订日期	修订说明	版本号	修订人	审核人
20160601	面向大数据的多源异构云计算平台	V0.1	吴恒	吴恒

目 录

1. 前言

1.1. 项目背景

云计算是创新 2.0 下的信息化系统发展的新业态，是知识社会创新 2.0 推动下的经济社会发展新形态，已成为促进信息技术产业发展和应用创新的主要推手之一，具有重要的国家战略意义。2015 年 1 月，国务院发布了《关于促进云计算创新发展培育信息产业新业态的意见》，国家明确云计算技术接近国际先进水平，构建自主标准体系目标；利用公共云计算服务资源开展百项互联网和大数据应用示范工程；引导传统企业的转型升级，鼓励积极拓展国际市场。

当前，云计算产业主要采用“以数据中心为基础、各自运营为手段、巨资竞争为策略”的实践模式，核心是集中式资源在线服务，通过资源规模化运营、广域共享和公用化服务实现 IT 总体实施成本降低和利用率提升，技术特征主要包括资源虚拟化、分配动态化、服务自动化等。例如 Amazon EC2、Microsoft Azure、Google Cloud Platform、阿里云、UCloud、金山云等。 随着以大数据为基础的智能计算技术深入发展，应用场景更加复杂，云计算强边界已经制约云应用的持续创新。为此，以多源异构云协作为基本特征、面向大数据的新型云计算模式逐渐成为学术界关注的热点和产业界关注的重点。

本项目为面向大数据的多源异构云计算平台，旨在通过多方云资源深度融合，以“软件定义”方式按需管理多源异构云；旨在通过软件优化、硬件加速等手段，满足大数据高效运行，便捷易用的目标。

1.2. 参考文献

1.http://www.cisco.com/c/en/us/products/cloud-systems-management/intercloud-fabric/index.html
2. http://dl.acm.org/citation.cfm?id=2904111&CFID=793855473&CFTOKEN=73457219

1.3. 术语与缩写

缩写、术语	解 释
SDN	软件定义网络，Software Defined Network
SDS	软件定义存储，Software Defined Storage
NFV	网络功能虚拟化、Network Function Virtualization
CI	持续集成、Continuous Integration
AC	应用编排，Application Compose

2. 需求分析

随着 IT 技术的快速发展和应用深入，信息化系统呈现出交付移动化、访问全域化、峰值极限化、增值精细化发展趋势，迫切需要企业 IT 架构的转型升级。企业 IT 架构是指机房硬件管理及信息系统开发交付模式，涉及信息系统如何运维，如何交付，如何开发三大核心问题。云计算是解决 IT 架构转型的主要技术途径，目前主要关注运维和交付问题，其中，虚拟机技术主要解决企业 IT 架构的高效能运维转型，核心是便捷、提高资源利用；容器技术主要解决企业 IT 架构的互联网交付转型，核心是敏捷、开发运维一体化。未来，以大数据为核心、具备按需分析、精准增值的应用将成为主流，如何应对大数据多源异构，协同复用、高效运行等特征，实现 IT 架构转型达到简化开发的目的面临挑战。

本项目基于云计算已有成果，通过整合多源异构云计算平台满足大数据多源异构的采集需要；通过引入数据湖满足大数据场景协同分析需求（http://www.vldb.org/pvldb/vol8/p1916-bhardwaj.pdf）；通过软件优化和硬件加速手段，实现云平台高效运行目标。具体而言，本项目主要解决企业 IT 架构的大数据开发转型，核心是快捷、简化开发复杂度。如图 1 所示，本项目是在整合虚拟机、容器平台基础上，重点支持数据湖场景。

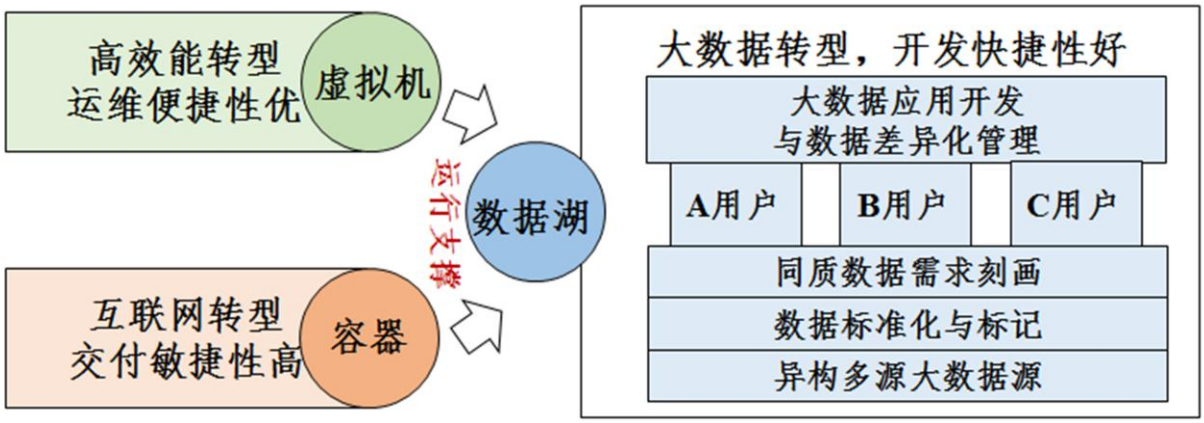

图 1 项目目标

数据湖场景是指不同人员以相同数据作为输入，差异化进行分析处理，得出个性化结果的过程。例如，某商业银行具有大量涉及消费的数据，通过对其标准化和分类，提供按需大数据供给服务。不同业务部门可能关心相同数据集(消费记录)，根据不同目标定制数据分析处理，得出消费年纪分布、或消费商品分布等多维结果。为应对该需求，开发通常采用数据拷贝，再验证

的方式，一般低效耗时。因此，迫切需要数据湖服务化和数据优化管理技术，如图 2 所示。

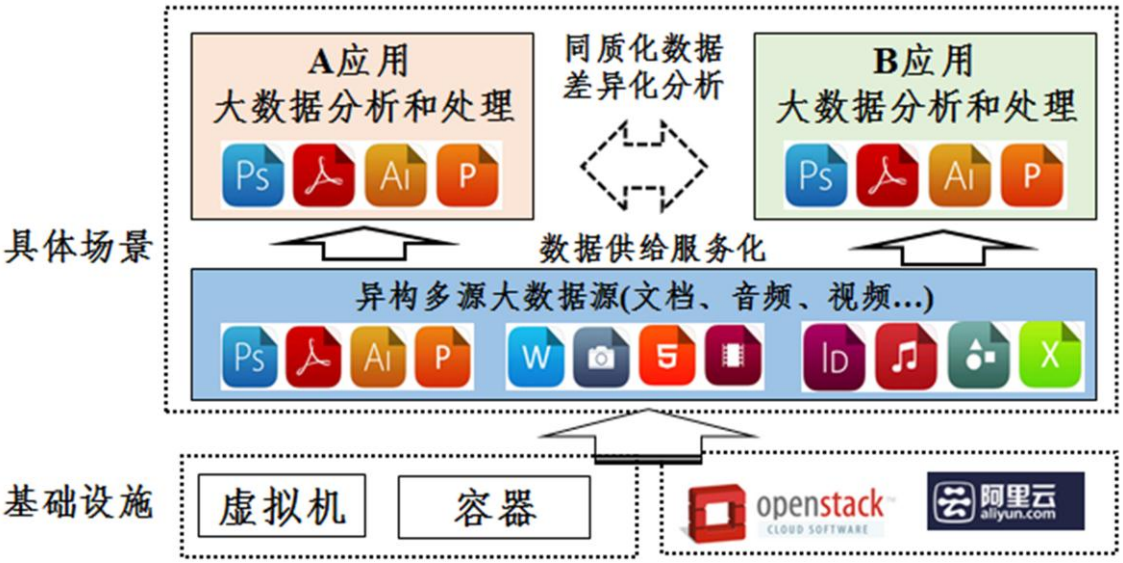

图 2 场景描述

3. 设计目标

3.1. 重要设计

1. 系统层面：集成多源异构云平台管理设计，至少可管理 Xen 和 Docker 两种异构虚拟化技术、至少具备 Aliyun、OpenStack 两种管理能力，提供服务化接口
2. 系统层面：集成配置中心设计，具备配置收集、变更通知、可靠运行等能力，提供配置组织模型、变更推送机制文档
3. 系统层面：研发物理机管理设计、具备生命周期管理、信息收集、核心软件安装和抽象能力，提供服务化接口
4. 系统层面：适配虚拟机/容器生命周期设计，提供服务化接口
5. 系统层面：基于虚拟机，实现 SDN、SDS、NFV 高级功能，提供服务化接口
6. 系统层面：基于容器，实现 CI、AC 高级功能，提供服务化接口
7. 研究层面：细粒度分布式应用追踪技术
8. 研究层面：自适应云应用性能预测技术
9. 研究层面：低开销性能干扰评估技术
10. 研究层面：多目标资源优化调度技术

11. 研究层面：高性能硬件加速技术
12. 研究层面：虚拟化操作系统构造技术
13. 研究层面：高可靠数据备份加强技术
14. 研究层面：数据协同管理技术

3.2. 设计原则

1. 多源异构云平台采用模块化设计
2. 服务化接口采用高安全设计
3. 设计异常和容错处理机制，参考异常处理文档

4. 架构设计

4.1. 总体架构

整体架构如图 3 所示，以服务注册与配置管理中心为核心，涉及物理机管理与软件安装、虚拟机管理与迁移优化、容器管理与安全加强、网络虚拟化及功能虚拟化、持续继与应用编排、权限管理与资源管控、界面框架与组装等模块。

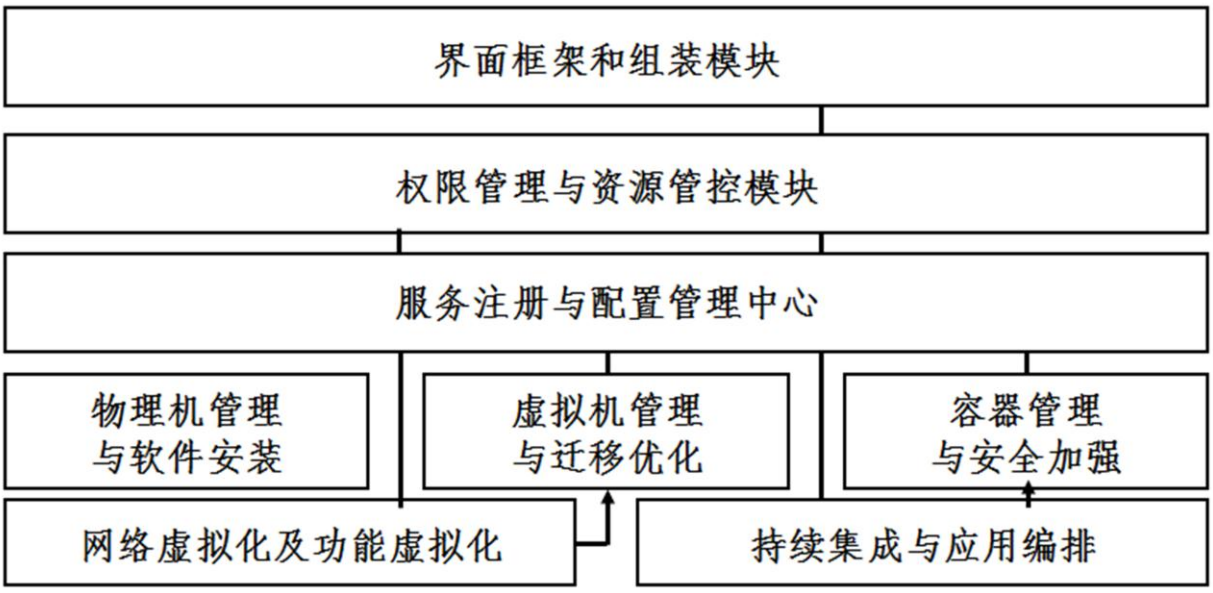

图 3 总体架构

- 服务注册与配置管理中心：用于统一记录其它模块的配置信息，维护配置变更及影响模块，保证其它各模块配置信息的一致性；

- 物理机管理与软件安装：具备物理机安装虚拟机/容器、配置网络、存储的能力，研发集群管理技术；
- 虚拟机管理与迁移优化：支持虚拟机生命周期管理，具备内存和硬盘在线迁移能力；
- 容器管理与应用加强：支持容器生命周期管理，强化其安全不足；
- 网络虚拟化与功能虚拟化：该模块有效地支持 vxlan、VPN 等能力；
- 持续集成与应用编排：具备应用从源码到容器的自动化生成，具备可视化应用组件刻画和一键部署能力；
- 界面框架与组装：所有模块的可视化操作。

4.2. 服务注册与配置管理中心

服务注册与配置管理中心是系统的核心，用于所有子系统配置的统一管理和变更推送。

4.2.1. 界面框架和组装模块

如图 4 所示，其它模块（权限管理与资源管控、物理主机管理和软件安装、虚拟机管理及迁移油画等）启动后，将 Url 信息注册到服务注册与配置管理中心，界面框架与组装模块周期性监听，并按需展示界面。

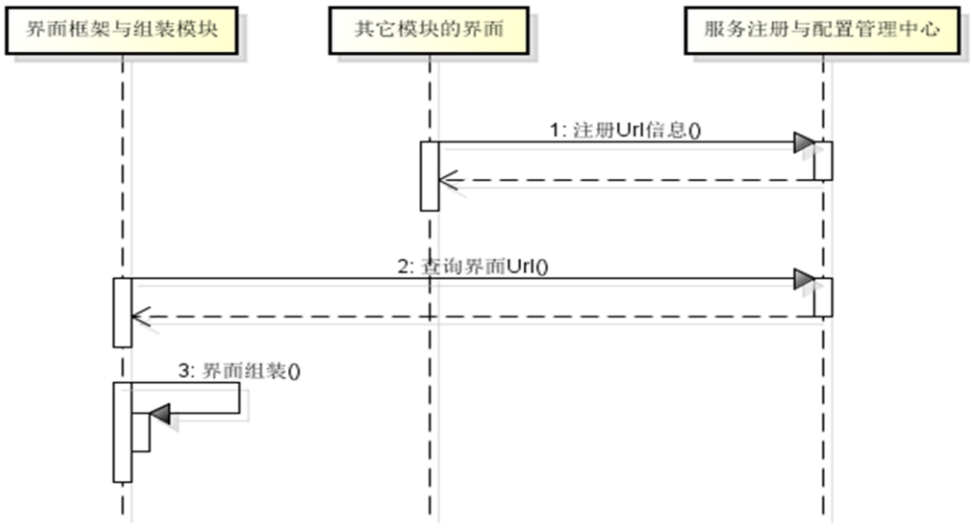

图 4 界面框架和组装模块时序图

4.2.2. 权限管理与资源管控

如图 5 所示，权限管理和资源管控模块启动后，将其自身 Url 信息注册到服务注册与配置管理中心，后续所有涉及权限的增删改查操作均持久化到数据库。

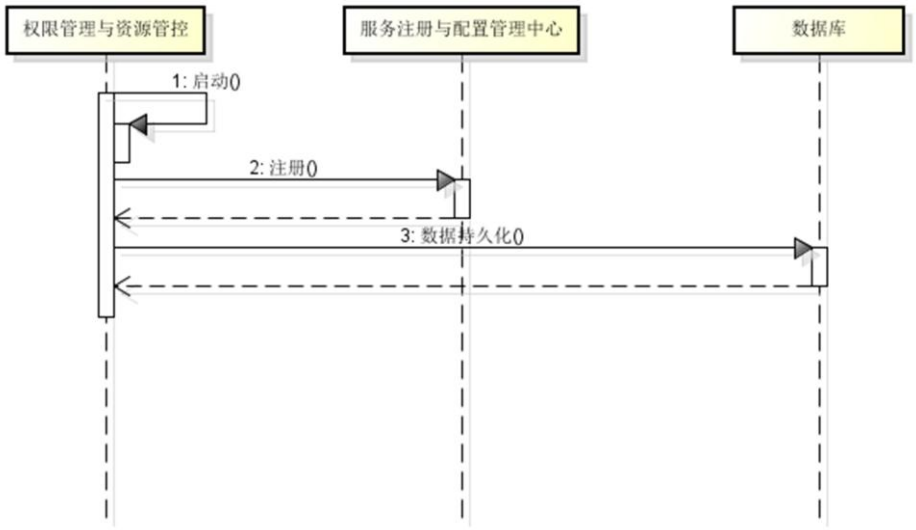

图 5 权限管理和资源管控时序图

4.2.3. 物理机管理与软件安装

如图 6 所示，物理机管理和软件安装模块启动后，将其自身 Url 信息注册到服务注册与配置管理中心，随后，服务注册与配置管理中心会周期性的监测该模块，判断其是否失效，具备健康检查能力。处于服务状态的物理机管理和软件安装模块，会提供相应的服务供外界调用。

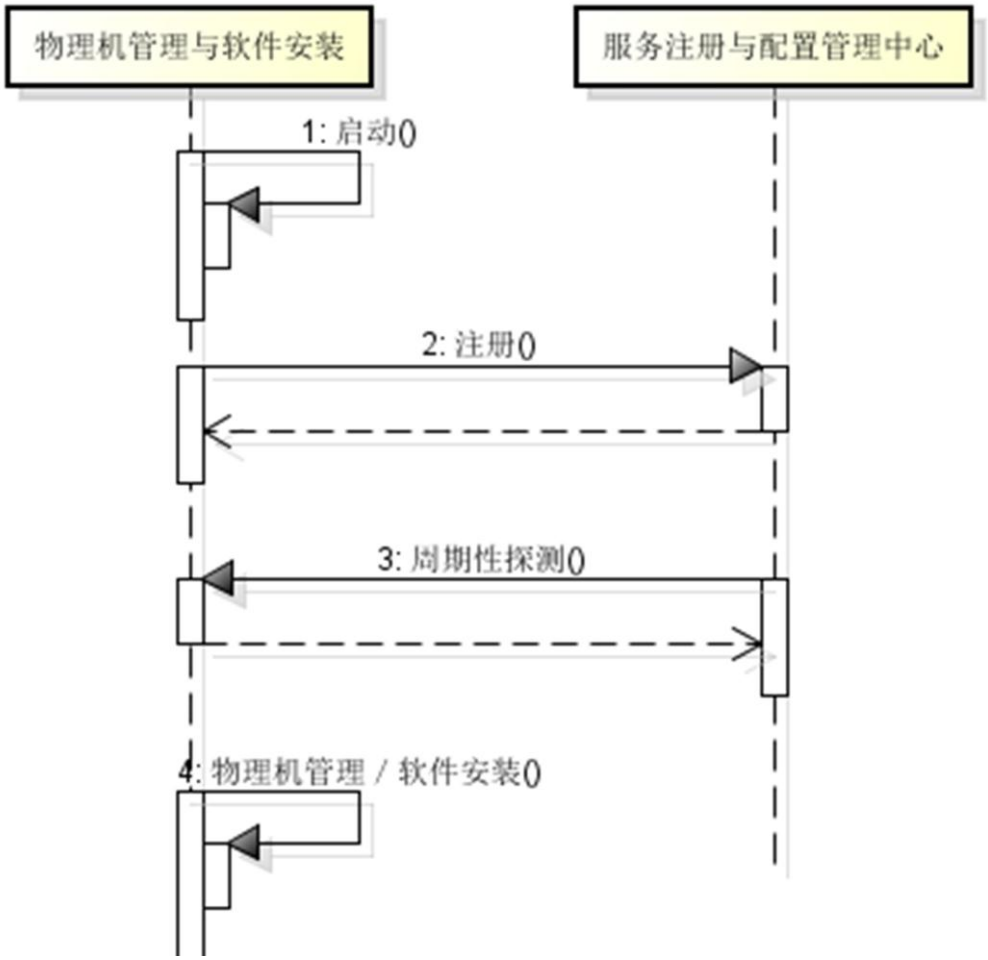

图 6 物理机管理和软件安装时序图

4.2.4. 虚拟机管理与迁移优化

如图 7 所示，虚拟机管理和迁移模块启动的前提条件是物理机管理和软件安装模块启动，该模块其自身 Url 信息注册到服务注册与配置管理中心，随后，服务注册与配置管理中心会周期性的监测该模块，判断其是否失效，其失效包括两种状态：(1)其依赖的物理机管理和软件安装模块失效；（2）其自身失效。处于服务状态的虚拟机管理和迁移优化模块，会提供相应的服务供外界调用。

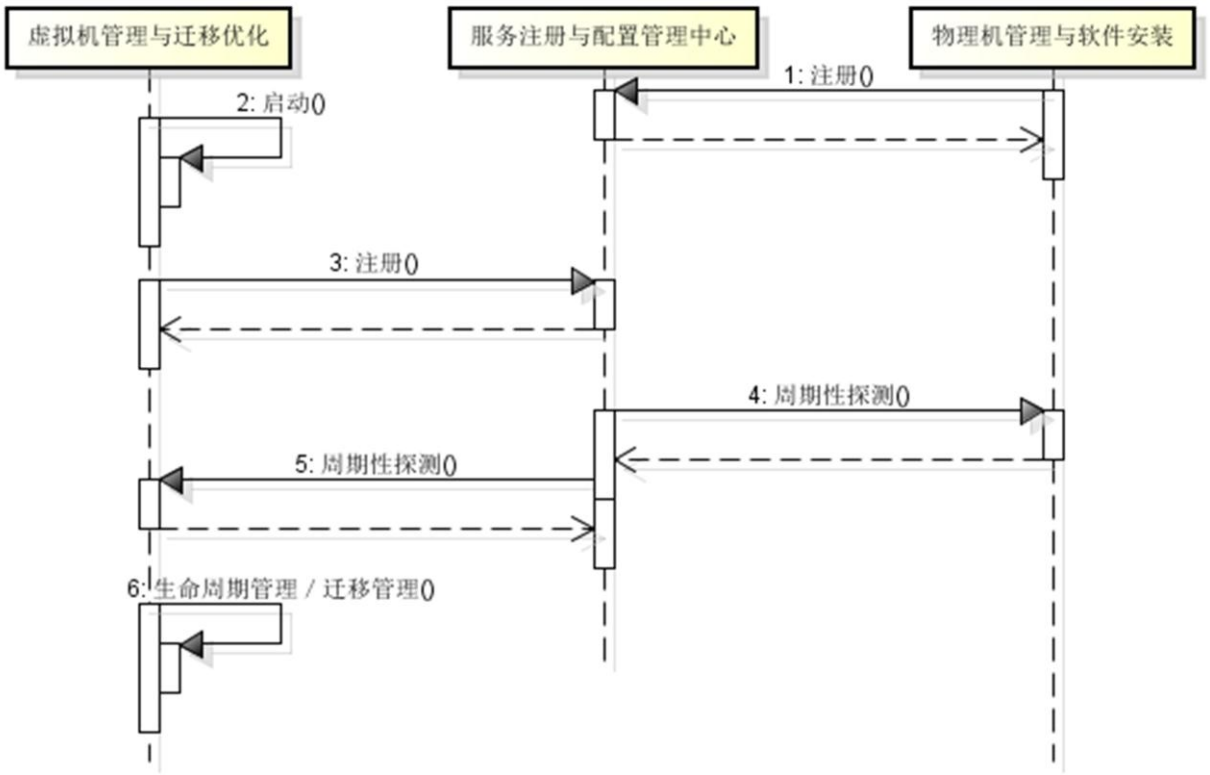

图 7 虚拟机管理和迁移优化模块时序图

4.2.5. 容器管理与安全加强

如图 8 所示，容器管理与安全加强模块启动的前提条件是物理机管理和软件安装模块启动，该模块其自身 Url 信息注册到服务注册与配置管理中心，随后，服务注册与配置管理中心会周期性的监测该模块，判断其是否失效，其失效包括两种状态：(1)其依赖的物理机管理和软件安装模块失效；（2）其自身失效。处于服务状态的容器管理与安全加强模块，会提供相应的服务供外界调用。

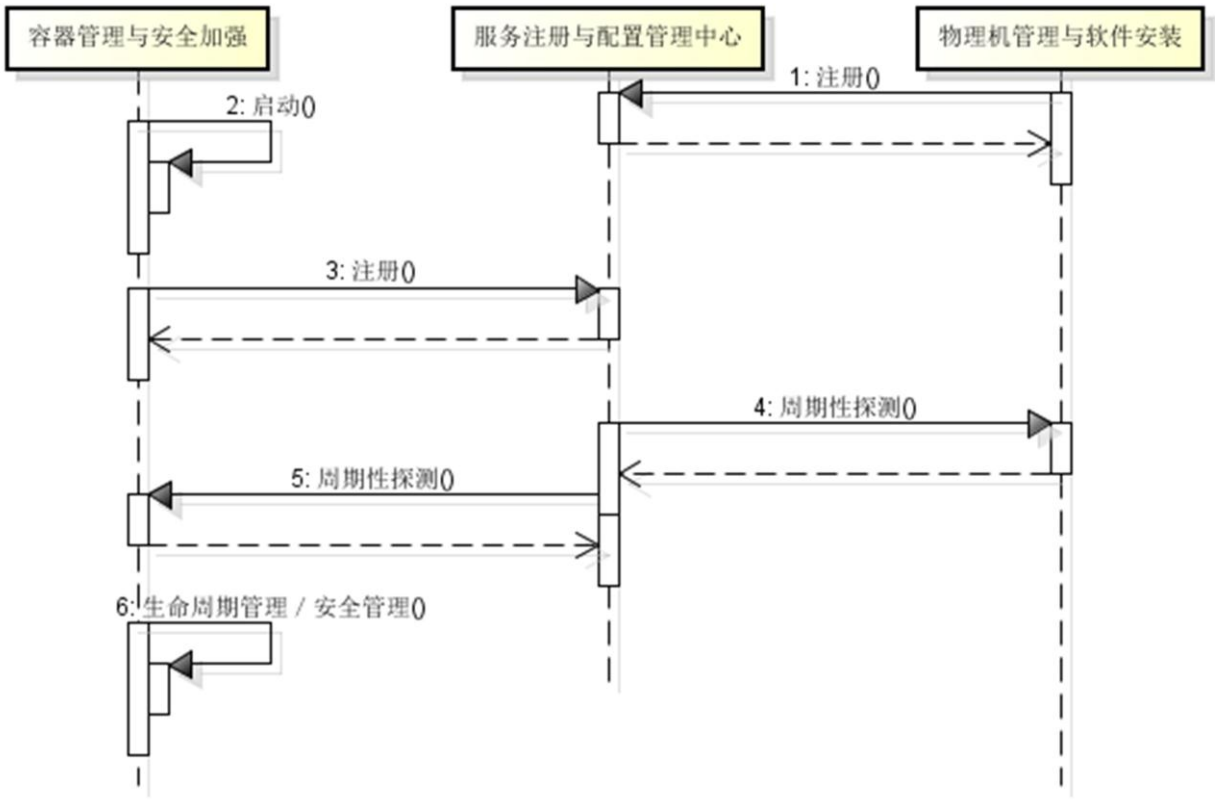

图 8 容器管理和安全加强模块时序图

4.2.6. 网络虚拟化及功能虚拟化

【后续补充】

4.2.7. 持续继承与应用编排

【后续补充】

4. 3. 设计策略

【如果有设计策略考虑，这里补充容量的设计思路
如果没有，标注“无”并说明理由】

4.3.1. 数据安全性设计

【如果有数据安全性设计考虑，这里补充容量的设计思路
如果没有，标注“无”并说明理由】

4.3.2. 扩展性设计

【如果有扩展性设计考虑，这里补充容量的设计思路

如果没有，标注“无”并说明理由】

4.3.3. 性能设计

【如果有性能设计考虑，这里补充性能的设计思路
如果没有，标注“无”并说明理由】

4.3.4. 运行安全设计

【如果运行安全方面的设计考虑，这里运行安全、系统容灾和系统容错的设计思路
如果没有，标注“无”并说明理由】

4.3.5. 容量设计

【如果有容量设计考虑，这里补充容量的设计思路
如果没有，标注“无”并说明理由】

4.3.6. 可维护性设计

【如果有可维护性设计考虑，这里补充容量的设计思路
如果没有，标注“无”并说明理由】

5. 核心组件调研

5.1. 服务组册与配置管理中心 Consul

Consul 是 HashiCorp 公司推出的开源工具，采用 Golang 语言实现，用于实现分布式系统的服务发现与配置。与其他分布式服务注册与发现的方案，Consul 内置了服务注册与发现框架、分布一致性协议实现、健康检查、Key/Value 存储、多数据中心方案，使用起来也较为简单。

与当前 Zookeeper、etcd 解决方案相比，该方案在性能、扩展性和多数据中心支持上具有优势：

对比项	Zookeeper	etcd	Consul
一致性	Paxos，复杂，效率不高	Raft，简单，性能好	Raft，简单，性能好
协议支持	TCP，集成支持 HTTP 等	HTTP	HTTP、DNS
健康检查框架	简单难扩展	不支持	易于扩展
多数据中心支持	不支持	不支持	支持

5.2. 多源异构云管理接口 libcloud

Libcloud 是一个 Python 库，于 2011 年 5 月成为 Apache 顶级项目，其目标是为各个云供应商的专有 API 提供了厂商中立的接口。其用户包括 Rackspace、Salit Stack、Scalr、DivvyCloud 、SixSq、CloudControl、mist.io、Cloudkick、GlobalRoute、Server Density、CollabNet。

与当前 jClouds 解决方案相比，在管理的广度和深度上均有优势：

对比项目	libcloud	jCloud
实现语言	Python	Java
支持公有云厂商（广度）	Amazon EC2、Azure Virtual machines、Google Compute Engine、SoftLayer、Aliyun ECS、CiscoCCS、DigitalOcean、Kili Public Cloud、Linode、vps.net、erverLove、Joyent、CloudSigma	Amazon EC2、Azure Virtual machines、Google Compute Engine、SoftLayer
支持私有云方案（广度）	Abiquo、CloudStack、Eucalyptus、OpenStack、VMware vSphere	Abiquo、CloudStack、OpenStack、
支持容器技术（广度）	支持	不支持
支持服务类型（深度）	ComputeService、BlobStore、LoadBalancer、CDN、DNS、Backup	ComputeService、BlobStore、LoadBalancer

后记：libcloud 对 Aliyun 支持存在缺陷，现已修复并提交社区 https://github.com/apache/libcloud/pull/800

5.3. 虚拟化管理接口 libvirt

libvirt 是一套免费、开源的支持 Linux 下主流虚拟化工具的 C 函数库。旨在为包括 Xen、kvm 在内的各种虚拟化工具提供一套方便、可靠的编程接口，支持与 C、C++、Ruby、Python、Java 等多种主流开发语言的绑定。

与当前 libxl、XenAPI 方案相比，开放性、扩展性与可控性优势明显：

对比项	libxl	XenAPI	libvirt
开发语言	C	Ocamel	C 语言，支持主流语言绑定

开源维护	Xen 社区	Ctrix 社区	虚拟化(KVM、Xen、Qemu..)社区
扩展性	与 Xen 代码耦合，扩展较难，升级较难	与 Xen 代码耦合，扩展较难，升级较难	与 Xen 代码松耦合，扩展容易
可控性	C 语言复杂	C 语言复杂	可选语言较多

5.4. 权限控制框架 Spring-Security

Spring Security 是一个能够为基于 Spring 的企业应用系统提供声明式的安全访问控制解决方案的安全框架。它提供了一组可以在 Spring 应用上下文中配置的 Bean，充分利用了 Spring IoC，DI（控制反转 Inversion of Control ,DI:Dependency Injection 依赖注入）和 AOP（面向切面编程）功能，为应用系统提供声明式的安全访问控制功能，减少了为企业系统安全控制编写大量重复代码的工作。

与当前 shiro 方案相比，使用的广泛性和社区活跃度较好。

对比项目	**Spring security**	**Apache shiro**
广泛性	认证、授权、加密	认证、授权、加密
活跃度	活跃	一般

5.5. 数据交换格式 JSON

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。它基于 ECMAScript 的一个子集。 JSON 采用完全独立于语言的文本格式，但是也使用了类似于 C 语言家族的习惯（包括 C、C++、C#、Java、JavaScript、Perl、Python 等）。这些特性使 JSON 成为理想的数据交换语言。 易于人阅读和编写，同时也易于机器解析和生成(一般用于提升网络传输速率)。

与当前 XML 方法相比，在简洁性和效率上具有明显优势。

6. 功能大类列表

6.1. 物理机管理与软件安装

- 支持通过 IPMI 管理物理机
- 提供 rpm 源和配置管理
- 安装和配置 Xen、Docker
- 网络配置，支持 OVS，可配置成 vxlan 和 dpdk 两种
- 存储配置，支持本地存储、OCFS2 和 Ceph 三种
- 获取物理机信息，包括系统信息，进程信息、服务等
- 监测物理机资源，包括 CPU、内存、网络和存储

6.2. 虚拟机生命周期管理与迁移优化

- 通过 ISO 安装虚拟机
- 虚拟机创建、启动、关机、删除，支持通过拷贝和模板两种策略创建虚拟机
- 虚拟机转变成模板、模板转换成虚拟机
- 虚拟机挂起、恢复
- 虚拟机内存快照和硬盘快照
- 虚拟机内存在线迁移和硬盘在线迁移
- 虚拟机登陆密码的定制化
- 虚拟机安全策略的个性化配置
- 虚拟机 CPU、内存、网卡和硬盘的在线变更
- 支持 VNC、Spice 两种协议交付访问虚拟机
- 虚拟机的集群管理
- 虚拟机的高可用管理

6.3. 容器生命周期管理与安全加强

- 容器仓库的安全配置管理

- 容器镜像的导入和导出
- 通过 Dockerfile 生成容器镜像
- 镜像的安全上传与下载
- 镜像的创建、停止和删除
- 容器的网络交付访问
- 容器 CPU、内存资源的在线调整
- 容器集群管理
- 容器高可用管理
- 容器内容安全检查

6.4. 网络虚拟化和网络功能虚拟化

- 支持 vxlan 的广播机制
- 支持 dpdk 加速机制
- 支持虚拟防火墙管理
- 支持虚拟路由
- 支持虚拟 VPN
- 支持内存模拟硬盘
- 支持 SSD 加速

6.5. 持续集成与应用编排

- 支持 git
- 支持 svn
- 支持持续集成工具
- 支持 Compose
- 支持 Nginx、Mysql、Redis、Sprak、HDFS 的容器化

6.6. 权限管理和资源管控

- 支持角色访问控制，控制用户能执行的操作

- 支持流程管理，满足个性化流程需求
- 支持资源隔离，控制用户可使用的资源

7. 外部接口需求

7.1. 硬件接口

【（1）系统是否存在需要调用外部设备 。】

【（2）如果没有需要注明“无”。】

7.2. 软件接口

【（1）与其它系统是否存在接口，具体描述。】

【（2）如果没有需要注明“无”。】

7.3. 其他接口

【（1）与其它系统是否存在接口，具体描述。】

【（2）如果没有需要注明“无”。】