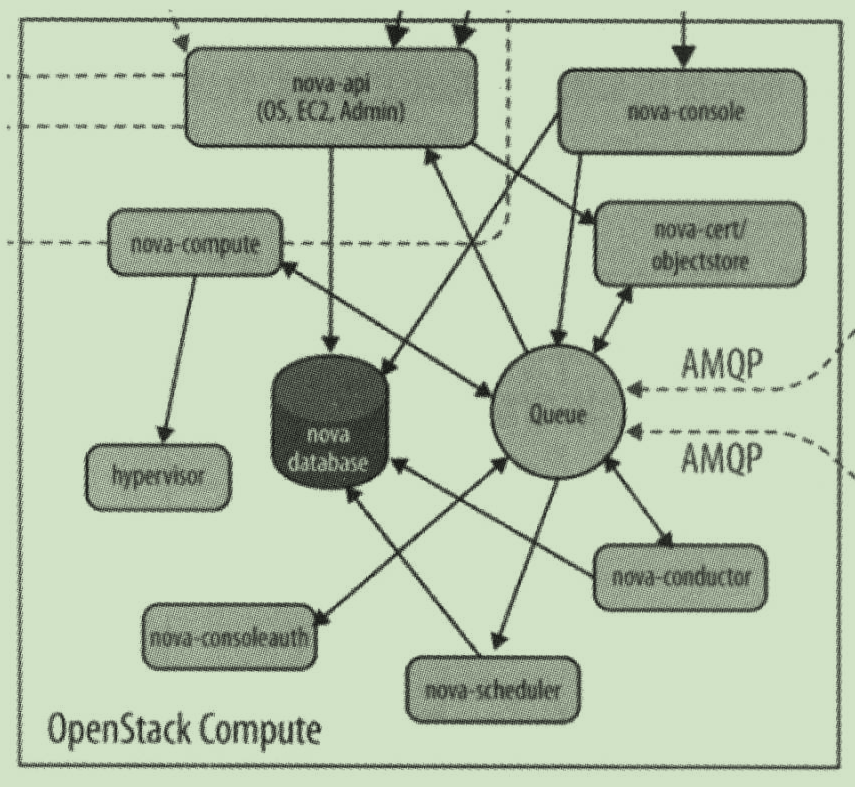
# 一．Nova服务

## （一）Nova架构

### 1.架构图

Nova的架构非常复杂，包含很多组件，这些组件以后台守护进程的方式运行。架构图如下所示：：



### 2.组件介绍

#### （1）API

##### a) nova-api

接受和响应客户的API调用，除了接受OpenStack自己的API，还支持Amazon EC2 API，也就是说nova-api兼容EC2 API。

#### （2）Compute Core

##### a) nova-scheduler

虚拟机调度服务，该组件负责决定应该在哪个计算节点上运行虚拟机。

##### b) nova-compute

该组件是管理虚拟机的核心服务，通过调用Hypervisor API实现虚拟机的生命周期管理。

##### c) Hypervisor

计算节点上跑的虚拟化管理程序，是虚拟机管理最底层的程序。不同的虚拟化技术提供自己的Hypervisor，常用的有KVM, Xen, VMware等。

##### d) nova-comductor

计算节点经常需要更新数据库，比如更新虚拟机的状态，处于安全性和伸缩性的考虑，nova-compute并不会直接访问数据库，而是将这个任务交给nova-conductor。

#### （3）Console Interface

##### a) nova-console

用户可以通过多种方式访问虚机的控制台：

nova-novncproxy:基于Web浏览器的VNC访问；

nova-spicehtml5proxy:基于HTML5浏览器的SPICE访问；

nova-xvpnvncproxy:基于Java客户端的VNC访问；

##### b) nova-consoleauth

该组件负责对访问虚机控制台的请求进行Token认证。

##### c) nova-cert

提供x509证书支持。

#### （4）Database

Nova有一些数据需要存放在数据库中，一般使用Mysql。数据库一般安装在控制节点上，Nova服务使用的数据库名称为nova。

#### （5）Message Queue

由于Nova包含众多的子服务，这些子服务之间需要相互协调和通信，为了解耦各个子服务，Nova通过Message Queue作为子服务的信息中转站，默认是RabbitMQ。

### 3.nova组件的物理部署方案

Nova的组件会部署在两类节点上：计算节点和控制节点。

计算节点：Hypervisor（运行着虚拟机）以及nova-compute；

控制节点：nova-scheduler,nova- conductor, nova-consleauth, nova-cert, nova-api, nova-novncpoxy, nova-compute, my-server（消息队列）, mysqld（Mysql服务端）

### 4. nova子服务的协同工作

（1）客户向nova-api发送请求，请求创建虚拟机；

（2）nova-api对请求作出处理后，向mq发送消息“让scheduler创建虚机”；

（3）nova-scheduler从mq中获取到API发送给它的消息，然后执行调度算法，从若干个计算节点中选出节点A；

（4）noav-scheduler向mq发送消息“在节点A上创建虚拟”；

（5）计算节点A的nova-compute从mq中获取到nova-scheduler发给它的消息，然后在本节点的Hypervisor上启动虚拟机；

（6）在虚拟机创建的过程中，计算节点A的nova-compute如果需要查询或更新数据库消息，会通过mq向nova-conductor发送消息，nova-conductor负责数据库访问。

## （二）Nova组件介绍

1.nova-api

nova-api是整个Nova组件的门户，所有对Nova的请求都首先经由nova-api处理。nova-api向外暴露若干个HTTP REST API的接口。在Keystone中我们可以查询nova-api的endpoints，客户端可以将请求发送到endpoints指定的地址，向nova-api请求操作。只要是和虚拟机生命周期相关的操作，nova-api都可以响应。

2.nova-scheduler

（1）flavor

创建Instance时，用户会提出资源需求，如CPU，内存，磁盘各需要多少，OpenStack会将这些需求定义在flavor中，用户只需要指定使用哪个flavor就可以了。Flavor中定义了VCPU，RAM，DISK和Metadata这四类细节，nova-scheduler会按照flavor来选择合适的计算节点。

（2）nova-scheduler的参数配置

在/etc/nova/nova.conf中，nova通过schedulerdriver, scheduleravailable\_filters和schedulerdefaultfilters这三个参数来配置nova-scheduler。

（3）Filter Scheduler

Filter Scheduler是nova-scheduler默认的调度器，调度过程分为两步：

通过过滤器filter选择满足条件的计算节点（nova-compute）；

通过权重计算（weighting）选择最大权重的计算节点上创建Instance；

Nova允许使用第三方的scheduler，配置scheduler\_driver即可。

（4）Filter

当Filter scheduler需要进行调度操作时，会让filter对计算节点进行判断，返回True或False。nova.conf中的scheduleravaiablefilters选项用于配置scheduler可用的filter，默认所有nova自带的filter都可以用于过滤操作。

另一个选项schedulerdefaultfilters，用于指定scheduler真正使用的filter。有如下的filter：

1. RetryFilter

作用是刷掉之前已经调度过的节点。这种过滤器是为了防止一个被分配的计算节点操作失败，通过RetryFilter可以直接过滤掉失败的节点，避免操作再次失败。

1. AvailabilityZoneFilter

为了提高容灾性和提供隔离服务，可以将计算节点划分到不同的Availability Zone中。OpenStack默认有一个Nova的Availability Zone，所有的计算节点初始都放在Nova空间中。创建Instance时，可以指定将Instance部署到指定的Availability Zone中。

AvailabilityZoneFilter会将不属于指定Availability Zone的计算节点过滤掉。

1. RAMFilter

RamFilter将不能满足flavor内存需求的计算节点过滤掉。注意，为了提高系统的资源使用率，计算节点的可用内存是允许overcommit的，即可以超过实际内存大小，nova.conf中通过ramallocationratio控制，默认值是1.5。

1. DiskFilter

DiskFilter将不能满足flavor磁盘需求的计算节点过滤掉。同样磁盘也允许overcommit，通过nova.conf中的diskallocationratio控制，默认值是1。

e）CoreFilter

CoreFilter将不能满足flavor VCPU需求的计算节点过滤器，VCPU同样允许overcommit，通过nova.conf的cpuallocationration控制，默认值是16。

f）ComputeFilter

ComputeFilter保证只有nova-compute服务正常工作的计算节点，才能够被nova-scheduler调度。

g）ComputeCapabilitiesFilter

该过滤器根据计算节点的特性来筛选，如计算节点的架构是X86还是ARM的，这些都是通过该过滤器过滤的。

h）ImagePropertiesFilter

该过滤器根据所选Image的属性来筛选匹配的计算节点，如Image指定运行在某种Hypervisor，这都是根据Image的Metadata来过滤的，如果Image没有设置Metadata，该过滤器就不会起作用。

（5）Weight

经过一堆过滤器的过滤后，nova-scheduler选出了能够部署instance的计算节点，再通过weight来对每个计算节点打分，打分最高的获胜。nova-scheduler默认的计算权重策略是通过计算节点空闲的内存量计算权重值，空闲内存越多，权重越大。

（6）scheduler日志

整个调度过程都记录在nova-scheduler日志中了，在/var/log/nova/scheduler.log中。如果要查询debug日志，就需要在/etc/nova/nova.conf中打开debug选项。

3.nova-compute

（1）Hypervisor

nova-compute在计算节点上运行，负责管理节点上的instance，nova-compute与Hypervisor一起实现OpenStack对instance生命周期的管理。nova-compute为不同的Hypervisor定义了统一的接口，Hypervisor只需要实现这些接口，就可以以Driver的形式直接插入到OpenStack系统中。

在计算节点nova-compute的配置文件/etc/nova/nova.conf中配置所对应的compute\_driver，如KVM配置的是Libvirt的driver。

（2）nova-compute功能

nova-compute的功能分为两类：

1. 定时向OpenStack报告计算节点的状态

要得到计算节点的资源使用情况，需要知道当前节点上所有instance的资源占用信息，这些都是通过Hypervisor的driver驱动获取instance的资源信息。

1. 实现instance生命周期的管理

OpenStack对instance的操作都是通过nova-compute实现的，包括instance的启动，停止，重启，暂停，恢复，终止，迁移，快照。

当nova-scheduler选定了部署instance的计算节点之后，会通过RabbitMQ向选定的计算节点发出启动实例的命令，该计算节点上的nova-compute收到消息后会执行instance创建操作。

（3）nova-compute创建instance的步骤

分为四步：

1. 为instance准备资源；

nova-compute首先会根据指定的flavor依次为instance分配内存，磁盘空间和VCPU和网络资源。

1. 创建instance的镜像文件；

资源准备好之后，nova-compute会为instance创建镜像文件。首先选择一个Glance中的Image，检查计算节点中是否已有该Image，如果没有则从Glance下载到计算节点中，然后将其作为backing file创建instance的镜像文件。镜像文件是通过qemu-img命令从Image中创建的，注意，镜像文件是instance启动盘对应的文件，而Image是Glance上保存的模板，即instance运行的模板。

1. 创建instance的XML定义文件；

创建instance的XML定义文件。

1. 创建虚拟网络并启动虚拟机；

为instance创建虚拟网络设备之后，就可以启动instance了。

4.nova-conductor

由于nova-compte需要获取和更新数据库中instance的信息，但是nova-compute并不会直接访问数据库，而是通过nova-conductor实现数据库访问。这样能够保证数据库的访问安全，以及更好的系统伸缩性，对于高并发的数据库访问请求，可以通过配置nova-conductor集群来分摊访问压力。