

**Nova 物理部署方案**

前面大家已经看到 Nova 由很多子服务组成，同时我们也知道 OpenStack 是一个分布式系统，可以部署到若干节点上，那么接下来大家可能就会问： Nova 的这些服务在物理上应该如何部署呢？

对于 Nova，这些服务会部署在两类节点上：计算节点和控制节点。 计算节点上安装了 Hypervisor，上面运行虚拟机。 由此可知： 1. 只有 nova-compute 需要放在计算节点上。 2. 其他子服务则是放在控制节点上的。

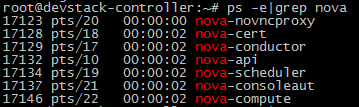
下面我们可以看看实验环境的具体部署情况。 通过在计算节点和控制节点上运行 ps -elf|grep nova 来查看运行的 nova 子服务

**计算节点**

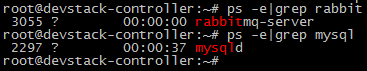
http://7xo6kd.com1.z0.glb.clouddn.com/upload-ueditor-image-20160421-1461203230379077302.jpg

计算节点 devstack-compute1 上只运行了 nova-compute 子服务

**控制节点**



控制节点 devstack-controller 上运行了若干 nova-\* 子服务

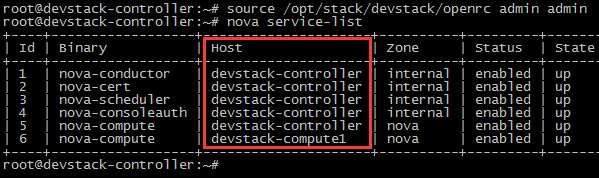


RabbitMQ 和 MySQL 也是放在控制节点上的

可能细心的同学已经发现我们的控制节点上也运行了 nova-compute。 这实际上也就意味着 devstack-controller 既是一个控制节点，同时也是一个计算节点，也可以在上面运行虚机。

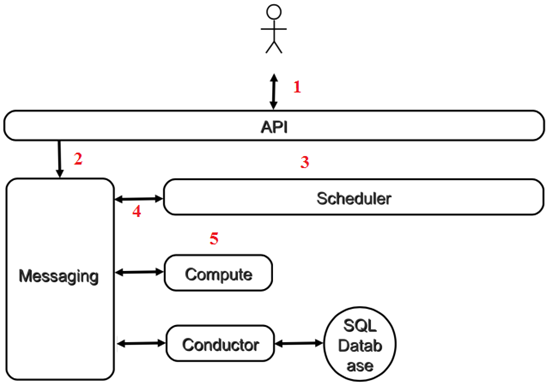
这也向我们展示了 OpenStack 这种分布式架构部署上的灵活性： 可以将所有服务都放在一台物理机上，作为一个 All-in-One 的测试环境； 也可以将服务部署在多台物理机上，获得更好的性能和高可用。

另外，也可以用 nova service-list 查看 nova-\* 子服务都分布在哪些节点上



**从虚机创建流程看 nova-\* 子服务如何协同工作**

从学习 Nova 的角度看，虚机创建是一个非常好的场景，涉及的 nova-\* 子服务很全，下面是流程图。



1. 客户（可以是 OpenStack 最终用户，也可以是其他程序）向 API（nova-api）发送请求：“帮我创建一个虚机”
2. API 对请求做一些必要处理后，向 Messaging（RabbitMQ）发送了一条消息：“让 Scheduler 创建一个虚机”
3. Scheduler（nova-scheduler）从 Messaging 获取到 API 发给它的消息，然后执行调度算法，从若干计算节点中选出节点 A
4. Scheduler 向 Messaging 发送了一条消息：“在计算节点 A 上创建这个虚机”
5. 计算节点 A 的 Compute（nova-compute）从 Messaging 中获取到 Scheduler 发给它的消息，然后在本节点的 Hypervisor 上启动虚机。
6. 在虚机创建的过程中，Compute 如果需要查询或更新数据库信息，会通过 Messaging 向 Conductor（nova-conductor）发送消息，Conductor 负责数据库访问。

上面是创建虚机最核心的几个步骤，当然也省略了很多细节，我们会在后面的章节详细讨论。 这几个步骤向我们展示了 nova-\* 子服务之间的协作的方式，也体现了 OpenStack 整个系统的分布式设计思想，掌握这种思想对我们深入理解 OpenStack 会非常有帮助。  
  
下一节我们将详细介绍 OpenStack 组件的通用设计思路。