数据库即服务的核心架构就是工作流引擎，这是实现自动化及按需服务的基础。  
1、生命周期管理  
生命周期管理功能包括数据库实例的新建、释放、扩缩容以及迁移。新建一个数据库实例包括分配资源(主要是主机资源)、安装数据库、初始化配置。对MongoDB来说，副本集涉及多个节点，涉及到资源的分配策略，Sharding更多。另外副本集还需要一些初始化工作，sharding需要有一个各组件的组合。释放实例比较简单，主要是资源的回收。扩缩容可以分为本地和跨机的扩缩容，其实跨机的扩缩容就是迁移。对于MongoDB来说，迁移可以直接利用MongoDB的添加节点自动同步的特性，还是比较方便的。  
生命周期管理功能主要涉及这几个组件，包括资源管理、规格及配置管理、软件栈管理和负载均衡:  
资源管理主要是指主机资源的管理，这里主机可以是物理机，也可以是虚拟机，如云主机等。资源管理主要负责资源的分配和回收，此外还包括如何实施资源隔离。  
规格及配置管理一个是需要为数据库实例制定一些规格，以方便扩容和缩容，另一方面是负责数据库相关配置的维护。  
软件栈管理则包括数据库软件以及其依赖的软件的安装维护等，包括操作系统。  
除了这些，还需要一个负载均衡组件来保证数据库实例在资源上的分布的均衡，当有主机资源需要下线的时候，能够做到自动对其上的数据库实例进行迁移。  
对于MongoDB来说，在实施资源分配策略时需要注意的一点是需要保证不要破坏副本集原本的高可用特性。虽然MongoDB副本集自带了高可用，但是如果你把副本集的所有节点都分布在一台物理机上，那如果这个物理机挂了，整个副本集都没用了。所以一个起码的原则是要保证MongoDB多副本的主机安全性，尽可能够做到机架安全。  
现在我们的MongoDB数据库即服务的架构可以稍微扩充一下了，多了资源服务、规格及配置服务、软件栈服务以及负载均衡服务这几个组件。  
2、容灾体系  
接下来看一下容灾体系，这包括高可用、备份/恢复、异地容灾/多活。高可用需要有一个负责健康检查的巡检服务，另外还需要有一个容灾切换的组件。备份/恢复也是容灾体系非常重要的一环，这里有一个很容易被忽视的事情是需要做备份的有效性验证。如果备份不是有效的，那等于没有备份。异地容灾/多活是比较高级的容灾能力，实施起来比较复杂，有兴趣的同学可以参考我之前做过的一个分享《MongoDB异地容灾多活实践》。  
(链接:https://yq.aliyun.com/articles/96598)  
MongoDB副本集自带了高可用，我们还需要做什么工作呢?主要是需要保证容灾切换的一个可控。以一个经典的3节点P/S/H副本集为例，一方面我们可以通过配置选举优先级的方式来保持Primary和Secondary的角色稳定性。另一方面，我们希望在任意时刻，用户都可以有两个节点是可访问的，因此我们需要对节点宕机后的副本集做一些reconfig操作，保证宕机节点最终都会变成Hidden，然后统一对Hidden进行处理，比如重搭等。  
容灾体系第二个比较重要的点就是备份恢复。备份主要需要做的是需要提供自动/手动的备份方式以及支持一些灵活的备份策略制定，如备份周期/备份保留时间等。恢复主要是看对恢复的形态做成什么样，是覆盖原来的实例还是克隆出一个新的实例来，还有就是恢复的粒度，这取决于备份能力，是只能恢复到某个全量备份，还是可以恢复到任意时间点。关于备份存储，我们要求的最要 能力是高可靠性。另外就是刚刚提过的备份有效性验证，不能等到火烧眉毛了才发现备份不可用，需要防范于未然。  
关于MongoDB的备份方法，相关的文档和分享已经有很多了，这里再简单提一下。全量备份从实施方式上可以分为两种，逻辑备份和物理备份。其中逻辑备份主要使用官方提供的mongodump/mongorestore工具。物理备份则可以在文件系统或是更底层的逻辑卷、块设备这层去做。  
从各个指标上对比逻辑备份和物理备份，在备份和恢复效率上，物理备份的优势比较明显，不过逻辑备份在兼容性上会比较好。  
MongoDB的增量备份主要通过持续的抓取oplog来实现。有了全量备份加增量备份，就可以实现恢复到任意时间点。  
至此，我们的MongoDB数据库即服务的架构又可以得到一个比较大的扩充，主要增加了高可用以及备份相关的一些服务。  
3、监控报警  
接下来看下数据库的监控报警，性能监控主要涉及性能数据的采集、存储和展示。采集粒度越细越好，最好能做到秒级。报警则可以分为可用性的报警和性能数据的报警。  
具备监控报警能力后的架构图已经有点满了，这里报警服务可以通过巡检服务和性能数据存储收集相关数据。  
4、增值服务  
来看最后一个增值服务，一个是审计，主要涉及审计日志的采集、存储和分析。另一个是诊断服务，一个是资源使用量上的诊断，另外一个是慢查询的诊断，可以做一些索引推荐等。  
这就是我们的MongoDB数据库即服务的完整架构，可以看到组件还是比较多的，做一个数据库即服务还不是那么容易的。  
总结  
最后做一下总结，我认为数据库即服务的核心特性有两点，一个是资源池化，另外一个是服务可量化。  
资源池化后才可以进行资源的自动管理，而我们需要的服务是要能够被量化的，并且是可控的。现在回顾一下之前的一键安装数据库，其实背后有许多工作要做。当然，如果觉得自己搭建一个数据库即服务太麻烦，可以考虑使用现成的云服务，比如阿里云MongoDB数据库服务:)  
Q&A  
Q1:这些阿里云上都有吗?还是内部再用?池化是多实例还是Docker ?  
A1:这些服务，现在阿里云上的云数据库MongoDB都做了，虽然有些功能可能还没有暴露在控制台上。多实例还是Docker，具体选择的技术，其实都可以的。  
Q2:想把MySQL的指定表数据存储到阿里云的MONGODB有好的实现方式么?  
A2:这个有一些工具可以干这个事情，可以关注阿里云的DTS服务，主要是在数据库中进行数据迁移，另外还有一个datax可以做异构数据库之间的导入导出。  
Q3:Mongo单个collection数据达到亿级别需要考虑拆分吗  
A3:亿级别可以考虑使用sharding了。  
Q4:它与Redis的区别主要是什么?  
A4:Redis更多可能还是缓存场景吧，MongoDB支持二级索引，可以支持很复杂的查询。  
Q5:测试插入200W数据，为什么各个分片分配不均衡呢，其中一个分片占比非常高。shard key 是递增的。  
A5:shard key递增就会导致在插入时都插入在一个shard上，发挥不了sharding的优势，可以考虑再组合一个字段作为shard key。  
直播链接  
https://m.qlchat.com/topic/details?topicId=270000444113427&isGuide=Y  
密码:221  
-AND-  
想了解更多数据库前沿技术?  
想聆听更多阿里的独门绝技?  
来Gdevops全球敏捷运维峰会北京站吧!  
9月15日 大牛亲授绝技 就差你了  
分享企业与嘉宾  
58到家高级技术总监 ||| 京东金融运维负责人  
当当网架构总监 ||| 饿了么技术总监  
前亚马逊中国区SDM ||| 新炬网络执行副总裁  
青岛航空高级架构师 ||| 润乾高级技术总监   
爱钱进DBA团队负责人 ||| 京东云数据库架构师  
滴滴出行云架构师 ||| 阿里云数据库开发负责人  
IBM ITSM技术专家 ||| 沃趣高级技术专家  
美团点评基础服务平台负责人 ||| 携程机票大数据平台Leader返回，查看更多  
责任编辑:数据库即服务的核心架构就是工作流引擎，这是实现自动化及按需服务的基础。  
1、生命周期管理  
生命周期管理功能包括数据库实例的新建、释放、扩缩容以及迁移。新建一个数据库实例包括分配资源(主要是主机资源)、安装数据库、初始化配置。对MongoDB来说，副本集涉及多个节点，涉及到资源的分配策略，Sharding更多。另外副本集还需要一些初始化工作，sharding需要有一个各组件的组合。释放实例比较简单，主要是资源的回收。扩缩容可以分为本地和跨机的扩缩容，其实跨机的扩缩容就是迁移。对于MongoDB来说，迁移可以直接利用MongoDB的添加节点自动同步的特性，还是比较方便的。  
生命周期管理功能主要涉及这几个组件，包括资源管理、规格及配置管理、软件栈管理和负载均衡:  
资源管理主要是指主机资源的管理，这里主机可以是物理机，也可以是虚拟机，如云主机等。资源管理主要负责资源的分配和回收，此外还包括如何实施资源隔离。  
规格及配置管理一个是需要为数据库实例制定一些规格，以方便扩容和缩容，另一方面是负责数据库相关配置的维护。  
软件栈管理则包括数据库软件以及其依赖的软件的安装维护等，包括操作系统。  
除了这些，还需要一个负载均衡组件来保证数据库实例在资源上的分布的均衡，当有主机资源需要下线的时候，能够做到自动对其上的数据库实例进行迁移。  
对于MongoDB来说，在实施资源分配策略时需要注意的一点是需要保证不要破坏副本集原本的高可用特性。虽然MongoDB副本集自带了高可用，但是如果你把副本集的所有节点都分布在一台物理机上，那如果这个物理机挂了，整个副本集都没用了。所以一个起码的原则是要保证MongoDB多副本的主机安全性，尽可能够做到机架安全。  
现在我们的MongoDB数据库即服务的架构可以稍微扩充一下了，多了资源服务、规格及配置服务、软件栈服务以及负载均衡服务这几个组件。  
2、容灾体系  
接下来看一下容灾体系，这包括高可用、备份/恢复、异地容灾/多活。高可用需要有一个负责健康检查的巡检服务，另外还需要有一个容灾切换的组件。备份/恢复也是容灾体系非常重要的一环，这里有一个很容易被忽视的事情是需要做备份的有效性验证。如果备份不是有效的，那等于没有备份。异地容灾/多活是比较高级的容灾能力，实施起来比较复杂，有兴趣的同学可以参考我之前做过的一个分享《MongoDB异地容灾多活实践》。  
(链接:https://yq.aliyun.com/articles/96598)  
MongoDB副本集自带了高可用，我们还需要做什么工作呢?主要是需要保证容灾切换的一个可控。以一个经典的3节点P/S/H副本集为例，一方面我们可以通过配置选举优先级的方式来保持Primary和Secondary的角色稳定性。另一方面，我们希望在任意时刻，用户都可以有两个节点是可访问的，因此我们需要对节点宕机后的副本集做一些reconfig操作，保证宕机节点最终都会变成Hidden，然后统一对Hidden进行处理，比如重搭等。  
容灾体系第二个比较重要的点就是备份恢复。备份主要需要做的是需要提供自动/手动的备份方式以及支持一些灵活的备份策略制定，如备份周期/备份保留时间等。恢复主要是看对恢复的形态做成什么样，是覆盖原来的实例还是克隆出一个新的实例来，还有就是恢复的粒度，这取决于备份能力，是只能恢复到某个全量备份，还是可以恢复到任意时间点。关于备份存储，我们要求的最要 能力是高可靠性。另外就是刚刚提过的备份有效性验证，不能等到火烧眉毛了才发现备份不可用，需要防范于未然。  
关于MongoDB的备份方法，相关的文档和分享已经有很多了，这里再简单提一下。全量备份从实施方式上可以分为两种，逻辑备份和物理备份。其中逻辑备份主要使用官方提供的mongodump/mongorestore工具。物理备份则可以在文件系统或是更底层的逻辑卷、块设备这层去做。  
从各个指标上对比逻辑备份和物理备份，在备份和恢复效率上，物理备份的优势比较明显，不过逻辑备份在兼容性上会比较好。  
MongoDB的增量备份主要通过持续的抓取oplog来实现。有了全量备份加增量备份，就可以实现恢复到任意时间点。  
至此，我们的MongoDB数据库即服务的架构又可以得到一个比较大的扩充，主要增加了高可用以及备份相关的一些服务。  
3、监控报警  
接下来看下数据库的监控报警，性能监控主要涉及性能数据的采集、存储和展示。采集粒度越细越好，最好能做到秒级。报警则可以分为可用性的报警和性能数据的报警。  
具备监控报警能力后的架构图已经有点满了，这里报警服务可以通过巡检服务和性能数据存储收集相关数据。  
4、增值服务  
来看最后一个增值服务，一个是审计，主要涉及审计日志的采集、存储和分析。另一个是诊断服务，一个是资源使用量上的诊断，另外一个是慢查询的诊断，可以做一些索引推荐等。  
这就是我们的MongoDB数据库即服务的完整架构，可以看到组件还是比较多的，做一个数据库即服务还不是那么容易的。  
总结  
最后做一下总结，我认为数据库即服务的核心特性有两点，一个是资源池化，另外一个是服务可量化。  
资源池化后才可以进行资源的自动管理，而我们需要的服务是要能够被量化的，并且是可控的。现在回顾一下之前的一键安装数据库，其实背后有许多工作要做。当然，如果觉得自己搭建一个数据库即服务太麻烦，可以考虑使用现成的云服务，比如阿里云MongoDB数据库服务:)  
Q&A  
Q1:这些阿里云上都有吗?还是内部再用?池化是多实例还是Docker ?  
A1:这些服务，现在阿里云上的云数据库MongoDB都做了，虽然有些功能可能还没有暴露在控制台上。多实例还是Docker，具体选择的技术，其实都可以的。  
Q2:想把MySQL的指定表数据存储到阿里云的MONGODB有好的实现方式么?  
A2:这个有一些工具可以干这个事情，可以关注阿里云的DTS服务，主要是在数据库中进行数据迁移，另外还有一个datax可以做异构数据库之间的导入导出。  
Q3:Mongo单个collection数据达到亿级别需要考虑拆分吗  
A3:亿级别可以考虑使用sharding了。  
Q4:它与Redis的区别主要是什么?  
A4:Redis更多可能还是缓存场景吧，MongoDB支持二级索引，可以支持很复杂的查询。  
Q5:测试插入200W数据，为什么各个分片分配不均衡呢，其中一个分片占比非常高。shard key 是递增的。  
A5:shard key递增就会导致在插入时都插入在一个shard上，发挥不了sharding的优势，可以考虑再组合一个字段作为shard key。