**Ambari接管线上Hadoop游戏数据集群实践**

**生产环境中Hadoop集群现状**

首先介绍我们生产环境中Hadoop集群的现状，Hadoop集群主要承担了数据接入存储、离线计算的职责，同时提供其上数据调度等自研系统的基础服务。生产环境中Hadoop使用的版本是v2.7.3，下面介绍其主要组件。

首先HDFS采用了HA with QJM的高可用架构，即采用Standby Namenode热备、多节点协同同步Active/Standby Namenodes(不同物理机器)之间元数据日志的方式，降低之前单点Namenode因为故障而导致的集群服务不可用的时间，提高集群的可用性。并且如下图中，Active/StandBy Namenode节点各自运行了基于Zookeeper集群自动监控Namenodes状态、自动选举保证集群Namenode只有一个处于Active状态的ZKFailoverController。



图1 HDFS HA with QJM

接着是用于集群资源分配与作业调度的Yarn框架，我们采用了Yarn的FairScheduler公平调度策略，并且根据不同业务线集群使用成本投入占比，划分了作业执行队列以及队列使用的资源池，各个资源池合理定义了最大同时运行作业数、Min/Max可参与分配的资源(Vcore数、mem大小)、动态竞争其他空闲队列资源池资源权重等参数，除开业务线使用的各队列之外，还定义了Default的共享作业队列，分配其较小的资源用于保证其他一些例如集群数据的共享查询等应用。



图2 YARN FairScheduler 资源分配图

集群ETL离线作业处理与数据查询我们主要基于Hive，Hive是SQL on Hadoop的数据仓库工具，具备良好的类似以关系型数据库的方式存储管理接入数据、提供对外数据处理接口特性，以Hive为基础我们打造了大数据中心，安全、简洁、便捷地统一接入数据以及对外统一提供数据处理基础服务。并且在发展过程中，为了提高数据中心服务水平，我们将Hive版本升级至v2.1，其HiveServer2服务能够更好支持并发和身份认证以及开放API客户端(如ODBC和JDBC)，且其更好支持内存计算(TEZ+LLAP)也为后续升级提供了基础。



图3 Hive 2.1 应用架构图

除开上述主要介绍的Hadoop集群中组件应用之外，我们还有包括Flume+Kafka分布式日志采集与队列数据分发的完整数据流架构，该架构使用Zookeeper集群做协同服务和配置管理，在这里不一一赘述。

最后是Hadoop集群原有的监控和告警逻辑，在进程监控这块我们采用Crontab周期性从主控节点ssh 至集群每个节点，轮询检查每个主要进程的运行状况，一旦发现进程挂掉则告警且重新启动；在性能监控这块(例如Yarn任务堆积数)我们也是采用Crontab周期性的调组件监控接口获取相关Mertrics 数值，告警异常值的方式。

至此我们已经梳理完毕生产环境中Hadoop集群现状，其上支撑的业务之广、使用人之多，就决定了我们Ambari线上接管集群不容有失，且需要最大限度降低接管过程中对业务造成的影响。

**Ambari相关技术介绍**

接下来我们将简单介绍Apache Ambari相关技术，Ambari是Hortonworks贡献给Apache开源管理Hadoop集群的顶级开源项目上文已提及，主要用于Hadoop集群的部署、监控与告警，Ambari整体架构如下图，主要有5部分组成：



图4 Ambari整体架构图

Ambari Web: 用户交互界面，通过HTTP发送使用Rest Api与Ambari Server进行交互。

Ambari Server: Web服务器，用于和Web、Agent进行交互并且包含了Agent的所有控制逻辑，Server产生的数据存储在DB中。

Ambari Agent: 守护进程，主要包含节点状态与执行结果信息汇报Server以及接受Server操作命令的两个消息队列。

Host: 安装实际大数据服务组件的物理机器，每台机器都有Ambari Agent服务与Metrcis Monitor守护进程服务。

Metrics Collector: 主要包括将Metrics Monitor汇报的监控信息存储到Hbase，以及提供给Ambari Server的查询接口。

Ambari整体管理集群方面以Ambari Server 为核心，维护着一个FSM有限状态机，包含平台中所有部署Agent并注册的节点、部署的服务与组件的状态变化信息、配置文件并且持久化在Ambari Server端的DB中。对外一方面通过restApi接口方式与Ambari Web交互，一方面接受来自Agent的定时心跳请求，所有交互信息中包含了节点状态、事件信息、动作命令中其中至少一种，由Ambari Server统一协调命令和维护状态结果，然后给Agent下发的相关command，Ambari Agent接受命令执行相关逻辑并返回状态结果。Ambari整体监控方面通过Ambari Server获取Ambari Metrics Collector中聚集后的从各个节点Ambari Metrics Monitor上报的单节点监控指标数据，在Ambari Web中给出图形化的展示。

Ambari是HDP数据平台套件的一部分，HDP是Ambari管理集群的技术栈基础。HDP即Hortonworks Data Platform，是Hortonworks完全开源以Yarn为核心整合Apache Hadoop技术的一个安全的企业级数据平台，HDP涵盖了几乎所有Hadoop的数据离线处理技术，以及最新的实时处理技术满足用户需求，如下图所示，其2017年开源的HDP v2.6正好支持Hadoop v2.7.3。



图5 HDP数据平台技术涵盖

Ambari支持对HDP的供应或者说Ambari基于HDP数据平台，下面是几个核心概念：

Stack：Ambari支持管理的HDP整个技术栈，本身技术栈也有版本区分，例如HDP 2.6 就是Hortonworks基于Apache Hadoop v2.7.3与Apache协议的2017年发行版。

Service：Ambari支持管理的某个具体服务，比如HDP中的HDFS、Yarn等，可以部署、管控的一个完整技术Framework技术方案。

Component：Ambari支持管理的最小组件单位，由于Service服务大多数为分布式应用，Componet即细分的Master、Slave、Client等组件。

Ambari按照Stack -> Service -> Component的层次关系，管理着HDP之间各组件依赖关系，通过Service Metainfo的定义来管理组件的依赖管理配置。例如YARN的metainfo.xml文件中定义了Yarn需要HDFS和MR2的支持，配置文件依赖Hadoop的主要配置文件core-site、hdfs-site等。其中HDFS、MAPREDUCE2为Ambari管理的Service，而HDFS中每一个运行实例例如Namenode、DataNode为Ambari管理的Component。

**Ambari接管线上Hadoop集群问题与思路**

上文中已提及在Ambari接管线上Hadoop集群时，需要考虑主要两方面的问题：

Ambari接管之后怎样保证集群依旧能够支撑原先支撑的所有上层应用；

Ambari接管动作怎样降低对线上Hadoop集群提供服务的影响。

**问题描述**

围绕上述这两方面，首先我们梳理了集群上支撑系统的所有使用接口，包括调度系统中使用的HDFS RestApi，数据中心使用的Hive Jdbc、ThriftApi，实时采集和分发系统使用的Zookeeper服务，其中涉及到的主要技术组件例如HDFS、Hive、Zookeeper，HDPHDP中各组件的版本应该向下兼容最好版本一致(Hadoop v2.7.3、Hive 2.1.0等等)保证功能特性满足。根据HDP提供的组件信息我们选择了HDP v2.6.0.2，并且针对HDP v2.6对其包含的组件功能特性与社区版各组件功能特性进行对比，确保了HDP v2.6.0.2支持现有所有功能特性。然后是支持HDP v2.6.0.2作为Stack的Ambari v2.5.1.0，接下来是确定Ambari v2.5.1.0管理HDP v2.6.0.2下的各个Service可行性：



图6 HDP-2.6.0.2与Hadoop2.7.3各Service版本对比

从上面的对比图中可以发现，Ambari是支持管理大多数与线上集群组件版本一致的组件的，但是也有例外例如Hive，Ambari支持管理的版本比线上集群中的低，而我们生产环境中数据中心等上层应用都是基于Hive 2.1.0接口运行的，所以Ambari接管线上集群不得不解决Hive的兼容性问题，才能最终达到Ambari管理为我们所用的生产集群的目标。接着需要考虑的是Ambari自动供应的HDP集群怎样与线上Hadoop集群兼容，就拿其中比较重要的一个必要条件来说：接管后的HDP HDFS能够唯一管理线上已有集群数据，而要能够顺利实现这一目标就得Ambari管理的HDP能够代替原有Hadoop集群，所以Ambari接管线上集群的问题就转化成了集群升级的问题，升级问题要考虑的是最小化停机的影响以及能够回滚恢复，以及充分利用Ambari的特性。

**解决思路**

现在已经了解到了此次Ambari接管线上集群的问题，我们来说说解决上述问题的思路。首先是Ambari管理组件兼容性问题，以Hive组件为例，虽然Ambari v2.5.1在管理方面只支持Hive v1.2.1，但是作为整个HDP v2.6.0.2 提供的技术组件中包含了Hive v2.1.0(见图5)，Hive v2.1.0组件的保留是为了兼容Hive2 包括LLAP、CBO等一系列新功能，在Ambari部署Hive组件时，会分为Hive v1.2.1与Hive v2.1.0两个版本同时部署，下图中通过主要jar包报名版本号可以看出，HDP功能组件根目录的hive与hive2子目录分别为Hive v1.2.1、Hive v2.1.0组件根目录。



图7 HDP中hive与hive2功能版本对比

Ambari部署Hive完毕后，接下来是生成配置与组件启动逻辑，而Ambari 默认是支持Hive v1.2.1即配置生成与启动时针对的是hive，那么我们需要调整的目标是让它针对hive2。回顾Ambari执行逻辑，完整的一次交互是用户在Ambari Web界面操作 -> Ambari Server请求处理&命令下发具体机器 -> 具体机器Ambari Agent执行相关操作，而在这里Ambari Agent启动Hive(实际分为HiveMetaStore、HiveServer、HiveClient等Component，每个Component启动执行逻辑一致，故在这里统一称为启动hive)过程包括了解析下发命令、从DB拉取配置信息、更新配置文件、启动Hive。因为配置文件对于各个Hive中Component一致，所以更新配置文件的执行逻辑也一致，并且hive2是能够向下兼容hive中的所有配置项的，因此在更新hive配置文件的逻辑最后增加同步配置至hive2配置的功能，就实现了在Ambari Web页面也能够更新本不支持的hive2的配置了，并且因为Ambari有自定义配置项功能支持所以也不用担心hive2配置项hive中不支持的问题。



图8 hive更新配置增加同步至hive2功能

配置支持的问题解决了然后是Hive启动的问题，启动逻辑更加简单即默认以hive/bin目录下启动脚本启动相关组件之前会尝试去获取机器的HIVE\_HOME系统环境变量，所以在节点提前配置hive2的相关系统环境变量，则启动逻辑会以hive2/bin目录下的启动脚本启动相关组件，同样的关闭、重启等逻辑也会按照hive2的逻辑执行。至此通过类似”移花接木”的方式实现了Ambari管理Hive v2.1.0的目标，即在Ambari Web页面上的管理操作的生效对象为HDP Hive2即Hive v2.1.0。

接着是Ambari接管线上集群如何转化成线上升级，集群升级中的要点包括：旧版本元数据的备份(Namenode、Journalnodes等)，旧版本配置在新版本配置的覆盖和调优，可能升级失败的回滚方案准备，实施升级时段选在集群使用空闲时间，升级前的演练等。这些方案此次Ambari接管线上集群升级同样受用，但是同时也要考虑一些特殊的细节：第一、Ambari支持的Hadoop集群供应并没有直接HA with QJM的方案，在HDFS部署时必须按照SNN冷备方案部署然后调整为HA with QJM的步骤；二、集群中的Zookeeper服务支持的一些实时型业务就决定了Zookeeper服务不能与Hadoop升级那样需要停机而造成服务不可用的空窗期；三、Ambari 管理的组件程序运行role与现有组件程序运行role的不同导致的主要包括文件权限等问题，以及由于此前Hadoop集群经历过节点扩容节点配置个性化差异如何在Ambari统一配置管理中避免冲突的问题。而且我们需要保证的目标包括了：一、架构上与原有集群一致，比如nn节点、dn节点的分布等，尽可能与原有集群组件机器分配一致；第二、升级最重要的是我们的数据资产，数据不能丢，所以重要配置比如hdfs各存储日志文件目录、索引文件目录等等必须一致；第三、再就是各组件重要参数，比如服务命名空间、文件块大小、资源调度策略等等，也要尽可能一致。所以综上所述，我们将Ambari接管线上集群升级拆分为下面几大步骤：



图9 Ambari接管线上集群升级步骤

在升级前的准备部分中将把所有的相关资源提前部署以及配置好，线上升级操作部分中只操作Ambari启动相关组件，完成线上集群运行组件的替换，所以集群停机影响时间缩短至Hadoop、Hive相关启动的时间，最大化减小了停机的影响。

升级前准备工作主要分为两个部分：第一、按照Ambari官网提供的部署方式部署并启动Ambari各组件，然后在Ambari Web上按照Ambari Cluster Install Wizard并且根据线上现有集群的组件分布，选择相应HDP组件的部署节点，确保将HDP各组件部署节点与线上集群各组件运行节点保持一致后，Ambari将会在各节点Install HDP的各组件，最后由于节点已有运行组件的端口冲突会导致Start 失败，不过不影响Ambari 成功完成部署HDP 各组件。

第二、Ambari线上升级前环境准备首先最重要的是与现有集群的配置同步，同样在Ambari Web界面中操作，这里需要注意的是现有集群不同节点的差异化配置在Ambari中使用Config Group同样进行差异化配置，比如集群中DataNode机器节点在磁盘上的差异情况，在Ambari中配置不同节点组对应的配置如下图：



图10 不同DataNode节点组Blocks目录差异化配置

依次对Hadoop、Hive等组件完成配置同步；接着是准备所有执行逻辑脚本，包括刚才提到的涉及到功能修改的相关Ambari Agent python功能脚本以及升级上线操作时的包括备份NN、JN元数据、统一修改系统环境变量等命令脚本，至此升级前的所有准备工作全部完成。

线上升级操作部分也分为两部分：第一、Ambari线上升级Hadoop，首先是Zookeeper集群的升级，采用从Zookeeper的follower机器开始一台一台停掉线上、Ambari启动相应节点，因为在配置同步过，所以Ambari 启动的zk是读取原有zk的数据，待所有follower节点操作完毕之后操作leader节点。然后是Hadoop的升级，Hadoop升级前暂时关闭所有程序访问入口(提前公告通知)，Hadoop升级中最重要的是Hdfs的升级，Hdfs的升级分为SNN冷备方案HDFS启动与Enable HA with QJM两步：第一步让原有集群进入安全模式确保没有数据写入时，备份所有Namenode、Journalnode节点元数据，执行关闭集群脚本在确保Hadoop组件全部关闭后执行修改所有节点系统环境变量脚本(修改为新HDP集群的系统环境变量)，根据Ambari中提示启动HDFS，由于Namenode重启需要一定时间(在这里不介绍Namenode重启优化了)，等待集群check blocks直至自动退出安全模式后至此SNN冷备方案的HDFS正式启动成功；第二步在Ambari中操作Enbale HA with QJM，每一步的操作根据操作指南即可，需要注意的是过程当中可能会出现原有的StandByNamenode元数据缺少因为第一步中单Namenode运行过程中产生的部分元数据，可以同步 Namenode 中fsimage与editLog文件至StandbyNamenode并重新initializeShareEdits，正常情况下在Enable HA 最后Ambari 又会重启一次HDFS，重启完成之后至此HDFS升级完毕，依次启动Mapreduce2、Yarn服务，整体Hadoop升级完毕。 第二、Ambari线上升级Hive，在第一部分Hadoop成功升级后相对来说Hive的升级比较容易，在更新Hive Metastore中相关元数据信息(DB SCHEMA)之前首先对数据库进行备份，更新Hive MetaStore Schema、执行启动Hive即可，因为已经部署了修改逻辑后的代码部分，Hive将以Hive v2.1.0在线上提供服务。

上面四大步骤顺利完成之后，Ambari就成功接管了线上集群，集群支撑的上层服务可以开放入口给使用者使用了。后续围绕Ambari的监控功能，使用其api接口可以定制各种个性化的监控和告警服务，至此Ambari成功接管线上集群。

**Ambari接管线上Hadoop集群实践**

上文中想必读者已经了解了我们Ambari实践最终想要达到的目标，其中存在的主要问题以及问题解决的思路，现在我们介绍Ambari接管线上Hadoop集群的实践过程。因为此次Ambari接管线上Hadoop集群动的是我们集群或者说是大数据平台的最底层，影响范围大所以每个步骤环节我们都谨小慎微确保无误，防止一丁点的疏忽导致整体大数据平台崩盘的”蝴蝶效应”出现。

所以我们从开发集群开始，目标是在使用Ambari从零搭建集群过程中，了解其核心特性和对机器运行环境的所有依赖；然后是在我们Hadoop测试集群上，Hadoop测试集群不仅有与生产Hadoop集群同样的环境与配置，并且也支撑着同样的用于测试目的的上层应用系统，在灰度集群上我们按照上文中四步骤完整演练了较多次，总结了一些其中碰见的问题和应急解决方案，除开节点规模和数据量比不上线上Hadoop集群之外，升级方案方面已经确定；最后是使用在测试集群中的上线方案，选择恰当的时机在线上环境完整执行了一遍，Ambari线上操作部分整体耗时控制在了2h以内，Ambari接管后集群整体运行正常，支撑的上层应用无报错。

下面将把Ambari接管线上Hadoop集群实践过程中的关键点着重讲述，让读者了解接管升级实践过程中需要关注的部分。

**配置同步**

Ambari接管的目标是对于集群使用者来说感受不到集群本身的变化，所以能否接管线上集群的关键点之一在于集群配置的同步，准确点是说Ambari使用的HDP中各个Component组件系统参数、应用参数、运行时环境变量等都需要保持一致，下面将介绍几个主要配置文件中的重要配置项。以下图中的HDFS core-site.xml配置为例，core-site.xml配置文件是HDFS重要配置文件之一，可以看见黄色标



图11 HDFS core-site.xml 主要配置项

注部分为我们在Ambari 线上升级Hadoop HDFS 时分阶段fs.defaultFS的不同配置，第一步单点启动时配置保证了单Namenode启动的顺利进行，到第二步Enable HA时，修改为HA时候的必要配置；然后在对于Hadoop集群使用者来说，增加了root、hive 用户与用户组的访问也是为了兼容Ambari接管后的HDFS运行role为hdfs用户的访问兼容，当然只有权限级别比较高的root和hive用户能够享有这个权限；接着是Namenode运行的最大内存大小等参数，Ambari默认的都比较小，这一点需要特别注意根据原有集群的运行Namenode进程运行时内存参数来调节这些参数。然后再来看看Yarn相关的配置项，我们以yarn-site.xml



图12 YARN yarn-site.xml 主要配置项

配置文件中主要配置项为例：橙色标注的部分为需要根据原有集群做调整的部分，其中包括需要根据具体节点内存大小情况来合理选择NodeManager可用于分配的最大内存，增加mapreduce\_shuffle选项以支持集群中的MapReduce程序，当然还有Yarn的调度策略，在生产集群环境介绍中已提到使用的是根据业务划分的FairScheduler，而Ambari中默认使用的CapacityScheduler，所以需要特别注意调度策略以及相关的配置与原有保持一致。

最后是一些组件的数据存储路径的配置，Ambari能够接管线上集群必须HDP各组件使用之前的数据来保证，所以数据存储路径的配置也是至关重要的。



图13 主要组件的主要数据存储路径配置项

**升级操作**



图14 升级操作具体步骤和细则

在所有升级前准备完成之后我们开始进行Ambari线上升级操作，线上升级操作分为两部分几大步骤如下执行：

在升级操作前我们会关闭所有访问集群的应用入口，特别是集群数据接入与定时作业这一块，保证集群没有数据继续写入后，我们接着关闭所有之前的集群进程运维监控脚本，防止升级过程中原有集群组件进程运行的恢复影响升级。接着为了方便在Namenode统一执行关闭集群操作，我们根据关停脚本的逻辑即各节点会找到存放对应进程(Datanode、NodeManager) PID文件路径，根据文件记录的PID执行Kill操作，但是由于部分PID文件存放在系统tmp文件夹下可能已经被删除，所以我们提前在各个节点部署了检查各自运行进程并还原可能缺失的PID文件的脚本，并且在正式关停前，统一执行还原进程PID文件，NN、JN节点元数据文件信息备份后才执行关停操作。然后在Ambari升级Hadoop之前我们通过Hdfs 管理界面记录系统的元数据信息，并且在Ambari 完成整体Hadoop升级后，我们详细对比了Hdfs 记录的元数据信息(下图中选择的为Ambari接管测试集群的统计数据)：



图15 Ambari接管测试集群升级前后文件对比

在确定升级前后Blocks完全一致后整体Hadoop升级确认完成。最后是Ambari升级Hive，在步骤2集群关停操作完成之后，我们同步进行了Hive MetaStore DB的备份，因为Hive MetaStore DB只存储Hive相关元数据信息，所以hivemeta DB本身不大，备份起来速度也较快。在正式Ambari Hive升级之前，使用SchemaTool工具更新hivemeta库，最后启动Hive。

我们在Hadoop与Hive升级启动完毕之后，我们迅速使用命令脚本模拟包括调度系统、数据中心等线上系统访问集群，测试新上线集群对外接口的可用性，确保测试都通过后，我们正式开放了各个上层应用，至此整体的上线时间耗时控制在了2h以内，整体升级操作时间轴如下：



图16 线上集群升级各过程时间轴

**后续运用**



Ambari接管线上集群后，在Ambari Web中可以对集群中所有组件进行统一管理：



图17 集群组件操作界面

上图中为集群中组件列表，以HDFS为例，对于HDFS的操作列表中包括了对NameNode、DataNode、JournalNode等的常用操作，通过界面即可统一操作。



图18 集群组件HDFS统一配置管理

统一配置管理将不同的配置文件分类给出，使用者根据相应的配置项属性名填写属性值同步即可，用户可自定义配置项，配置同步后有历史版本概念，多版本之间可以对比。

Ambari Web中将监控集群捕获到的关键操作指标值通过良好的Widget可视化，帮助我们日常能够快速排查和解决问题，主动预防问题上面提高了不少效率。例如我们通过观察Yarn Pending Apps的个数，发现堆积比较严重的时段后，会合理去调整作业的执行时间；通过观察集群整体CPU与内存的利用率，可以快速定位集群计算能力的瓶颈即CPU使用率较高但是内存使用率相对较低，然后我们会合理调整Yarn中Container对于CPU利用的策略等。



图19 监控指标值可视化

由于我们可以在统一的Ambari Web界面中对集群组件进行操作、配置管理，基于Ambari集群统一管理特性标准与规范化了集群的运维管理操作，相应的操作例如增加删除节点、组件迁移、配置修改等，都有明确的权限范围以及完整的操作历史记录，并且将Ambari 所有使用者入口做统一管理，公司内部相关人员只能在权限范围内对集群进行有迹可循的操作。



图20 集群管理者各角色权限表

如上表将对于集群的运维管理操作角色权限分成四个Level，最低的Level是能够在Ambari Web中查看集群的运行状态、配置、告警等信息，但是对于集群没有任何的操作权限，此类权限开放给所有需要对集群运行健康状态有关注的集群使用者，例如可以在Tez View中查看自己的作业运行情况，在Yarn 管理界面中查看负载等等。然后在针对集群可操作人群，同样划分了操作组件级别、集群整体级别两个Level，一般的集群运维人员可以去管理其中组件、进行调优，而上升到集群统筹规划这一层面，则需要对整体架构熟知的集群架构人员去管理，最后超级管理员除开上述所有权限之外，多了一层Ambari本身系统级别的管理。管理权限按层级划分，即能够满足不同集群使用者的要求，同样也保护了集群的运行安全和稳定。

然后我们根据Ambari提供的监控功能开发了相应的配套处理程序，目的在于第一使用我们自己的告警系统去替代Ambari中不太友好的告警系统；第二充分利用Ambari api不仅实现告警而且能够在故障出现时一定程度上尝试自动恢复。提高我们在集群监控、管理方面的效率。

首先Ambari提供了良好的Restapi用于与集群的各种直接交互，下面我列举一组Restapi示例：

http://ambari /api/v1 /clusters/hdp /hosts /${host\_ name}/ host\_components/ ZKFC

|  |
| --- |
| {"RequestInfo": {"context": "ReStart ZKFC","operation\_ level": {"cluster\_ name": "hdp","host\_ name": "${host\_ name}","service\_ name": "HDFS"}},"Body": {"HostRoles": {"state": "STARTED"}}} |

上述URL为典型的操作指定机器ZKFC进程的命令，如果Http动作是GET则返回该进程的状态信息，如果是PUT且增加json请求内容则是对ZKFC的一次具体操作，从RequestInfo中context的描述可以看出是对ZKFC的重启操作，operation\_level描述了具体操作对象的信息属于哪个集群哪个节点，以及ZKFC属于HDFS服务的一个组件，最后Body HostRoles描述了ZKFC重启操作后应该属于启动状态。

介绍完了Ambari中的操作API，我们利用api的特性设计了一套完整的自动发现组件疑似严重错误、确认错误并告警、尝试恢复的功能配件，完整逻辑图如下：



图21 利用Ambari操作集群api进行自动恢复逻辑

定期扫描dm\_monitor\_info Ambari中定义的告警项的周期扫描状态，发现组件存在的最近一次的隐患，确认组件是否真正处于服务不可用的状态(可能是集群在维护即maintainance\_state=‘ON’)后，记录该次告警信息，周期性尝试自动恢复正常之前告警系统报告消息，直到恢复后最后一次向告警系统报告成功恢复消息后，消除此次告警信息。

根据上线后运行近半年的统计，累计自动恢复组件时间分布如下图，Ambari中最小扫描周期为1分钟，所以按照分钟级别的扫描出疑似组件问题与自动恢复的平均时间在五分钟之内，且绝大多数故障恢复在一分钟，大大提高了组件的服务可用性。



图22 利用Ambari监控做自动故障恢复时间分布

**后记**

在Ambari接管线上集群后已经稳定运行了半年之久，它帮助我们大大提高了集群管理、监控方面的效率，在帮助我们性能排查、科学调优方面给了很大的帮助。当然Ambari管理与监控集群只是大数据平台基础建设的第一步，在智能化、企业级大数据平台基础建设过程中，我们会利用其提供的HDP平台服务不断提高大数据平台基础服务水平。