HBase hbck 分析报告

HBaseFsck（hbck）是HBase中用于region一致性和表完整性检查和修复的命令行工具，它有两种基本工作模式：只读不一致性识别模式和多阶段读写修复模式。hbck运行过程中会扫面.META.表以及HBase使用的hdfs的root目录（主要是.regioninfo文件）来收集其持有的所有相关信息，主要是元数据信息，之后比较收集到的信息来报告一致性和完整性问题，然后根据相应命令行参数执行修复操作。一致性检查以region为单位，它会检查region是否同时存在于.META.和HDFS中，通过比较.META.表中的数据和.regioninfo文件中的数据信息，并检查其是否只被指派给唯一的region服务器。完整性检查以表为单位，它将region与表细节信息来进行比较以找到缺失的region，同时也会检查region的起止键范围中的空洞或者重叠情况。

###### 一 HBCK内部执行流程

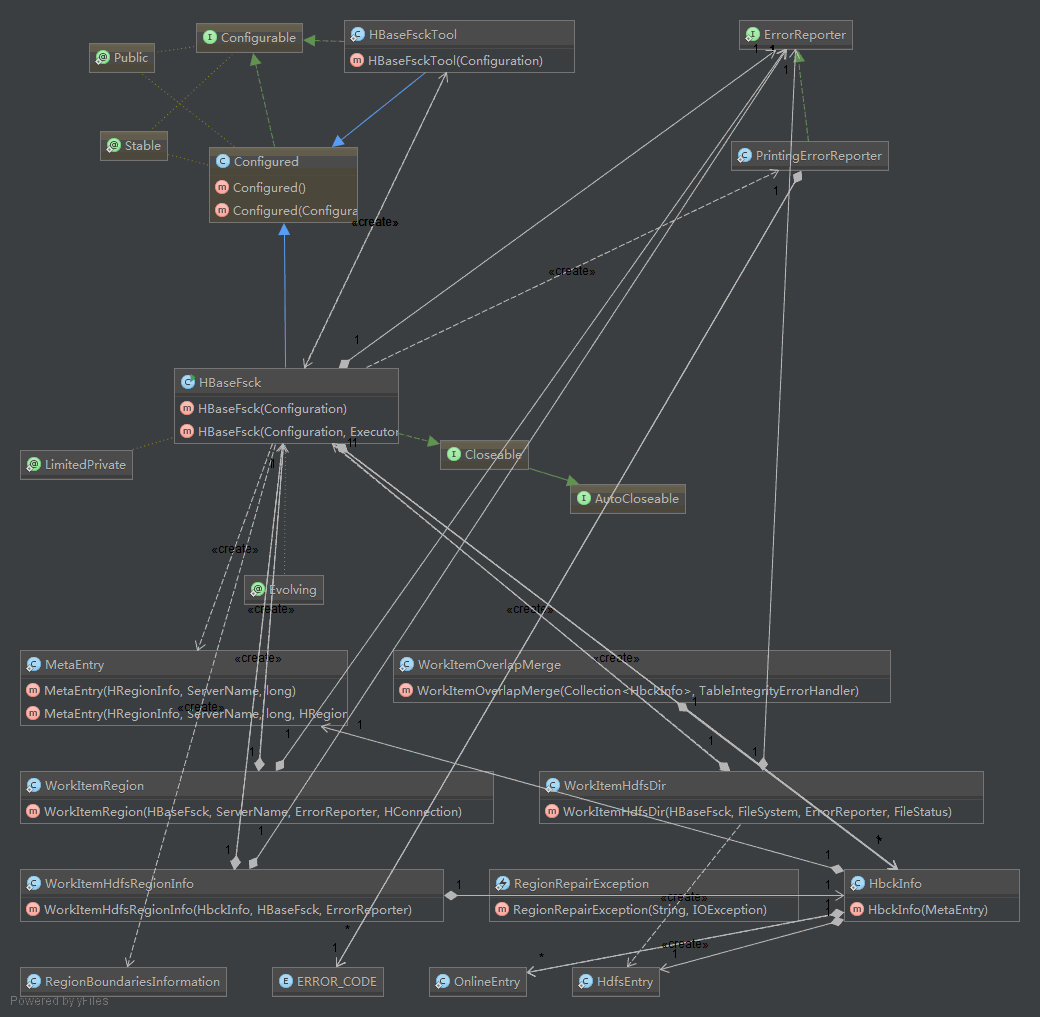
hbck作为检查和修复命令行工具其执行过程涉及到数据信息收集、问题检查、修复以及最终结果的报告的步骤，其中，数据信息收集、比较、问题确以及不带修复参数的结果报告可认为检查阶段，整个内部操作顺序大概分为如下三步：

1，检查不一致性和完整性问题；

2，如有存在表的完整性问题，则先修复该问题；

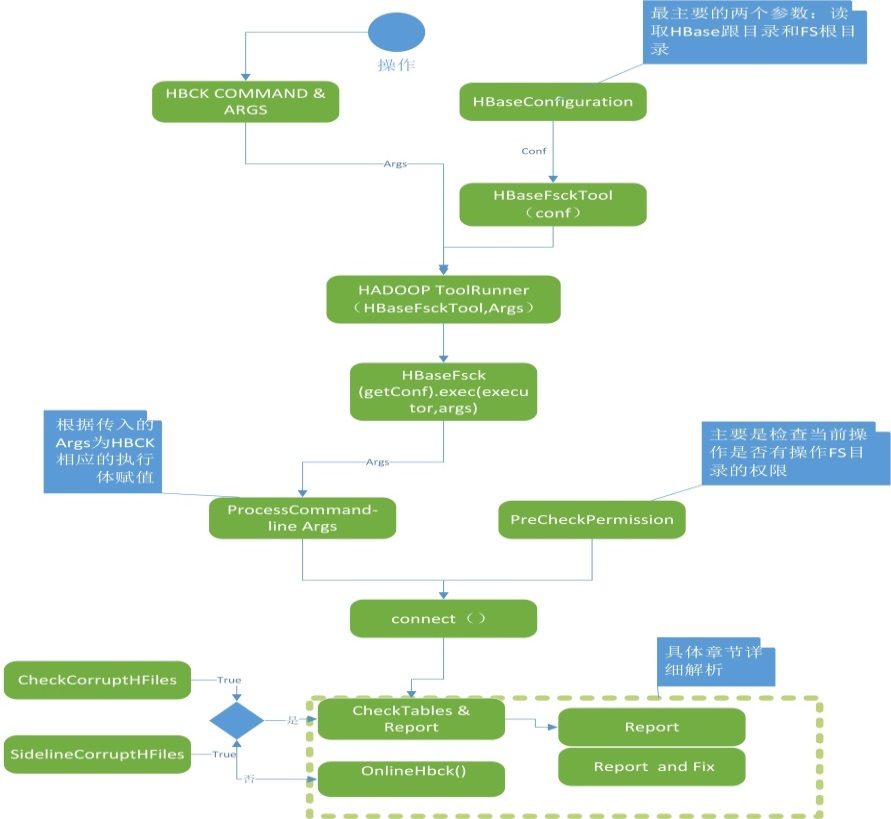
3，如有region级别的不一致性，则修复该级别的问题，在修复期间region处于关闭状态 。

hbck主体操作有HBaseFsck及其周边类如：HBaseFsckRepair、TableIntegrityErrorHandler等定义及实现

HBaseFsck类如图所示，根据其在执行过程中所涉及到的内外部文件和执行体调用，可具体分为以下几个步骤的重要操作：

1. 配置信息加载，由hbase-default.xml和hbase-site.xml配置，最主要是关于元数据和FS的相关配置（类中conf先关函数及接口）；
2. 命令行参数处理，根据命令行传入的参数确定最终执行的操作及其顺序；
3. 访问权限检查，主要是FS文件所在目录的访问权限；
4. 执行一致性与完成性检查命令；
5. 执行表完整性修复指令；
6. 执行region一致性修复执行
7. 报告结果。

其大概流程图如下所示：



###### 二 检查

一致性检查以region为单位，它会检查region是否同时存在于.META.和HDFS中，通过比较.META.表中的数据和.regioninfo文件中的数据信息，并检查其是否只被指派给唯一的region服务器。完整性检查以表为单位，它将region与表细节信息来进行比较以找到缺失的region，同时也会检查region的起止键范围中的空洞或者重叠情况。

检查命令:

*$ ./bin/hbase hbck*

如果需要查看更详细的信息可通过

*$ ./bin/hbase hbck –detail*

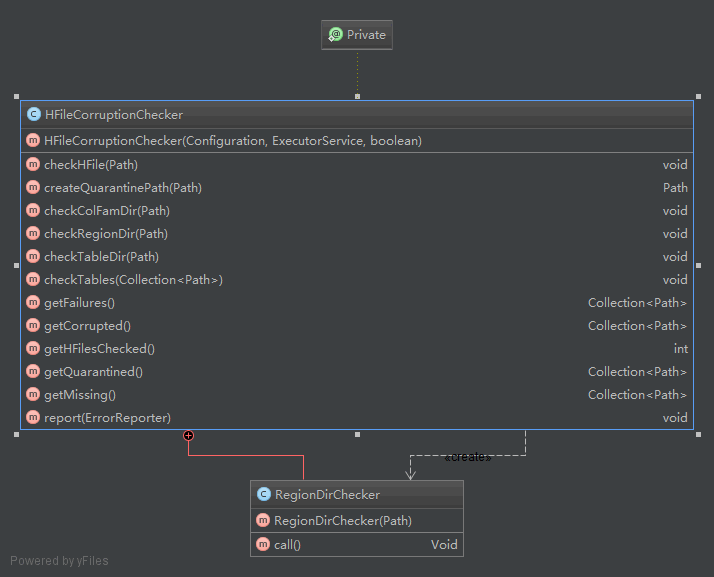
如果仅对指定的表进行检查可在hbck命令行后添加表名比如:仅检查TableFoo和TableBar可通过以下命令实现:

*$ ./bin/hbase hbck TableFoo TableBar（最终最合并到一个叫includedTables的集合）*

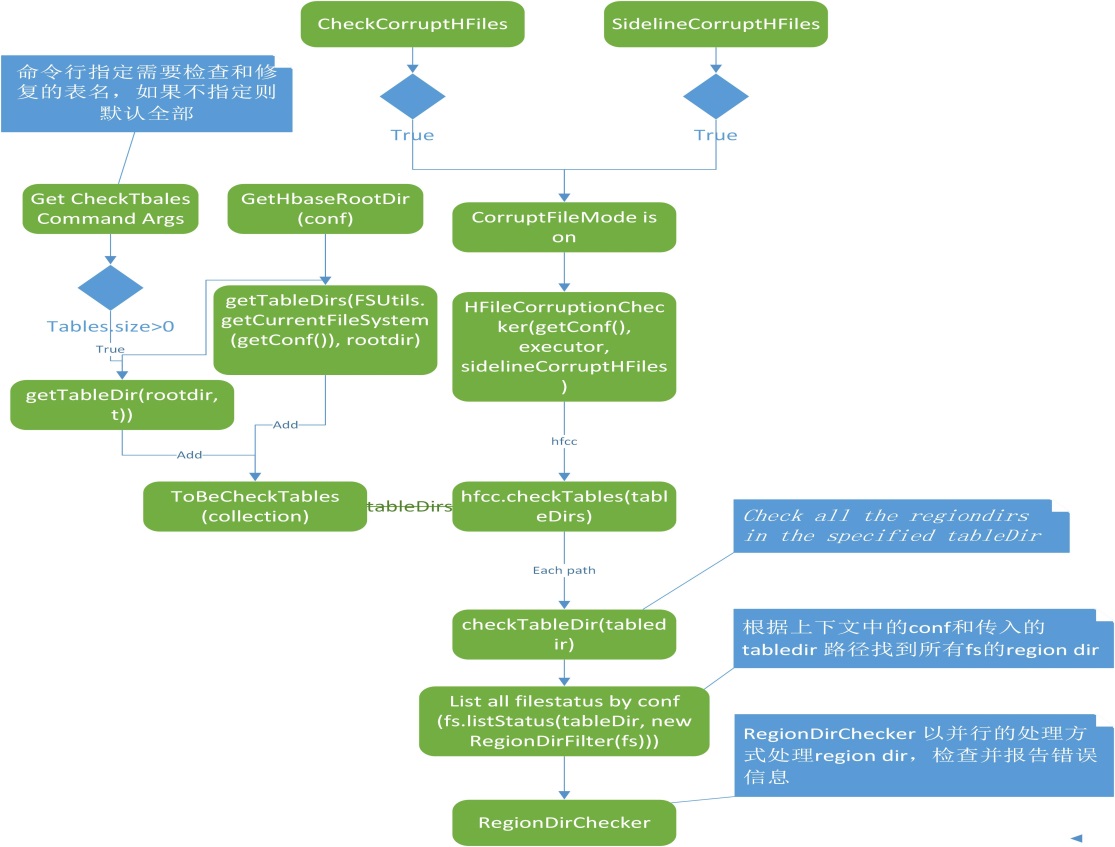
需要注意，在对region进行一致性修复时必须保证系统处于online状态，也就是说必须首先保证connect()函数被成功调用。

1. **HFile Corrupt检查**

如果在执行hbck时打开了HFiles Corrupt Check模式，即传递了"-checkCorruptHFiles"和"-sidelineCorruptHFiles"参数，则首先会进行HFiles检查，防止后续有操作打开那些存在问题的HFiles，该项检查由HFileCorruptionChecker完成，其类图如下所示：



根据类中的函数我们可以大概看出其所设计的检查包括：region、列簇、列、表等文件，其检查流程如下图所示：



首先，系统读取到"-checkCorruptHFiles"和"-sidelineCorruptHFiles"参数，判断是否需要进行HFiles Corrupt检查，如果需要会根据当前上下文中的conf、一个exector和布尔值sidelineCorruptHFiles构造一个HFileCorruptionChecker对象hfcc，并根据命令行传入的需要检查的表合并为需要检查的表集合includedTables，如果没有参数传入则默认所有表都进行该项检查；之后，根据conf的hbase.rootdir配置获取rootDir，root dir和 includedTables一块确定需要检查的table的路径信息，并以集合的方式返回；然后，调用hfcc的checktables函数作进一步操作，最终调用checkDir对region和table进行检查并返回检查的结果。

1. **表完整性检查**

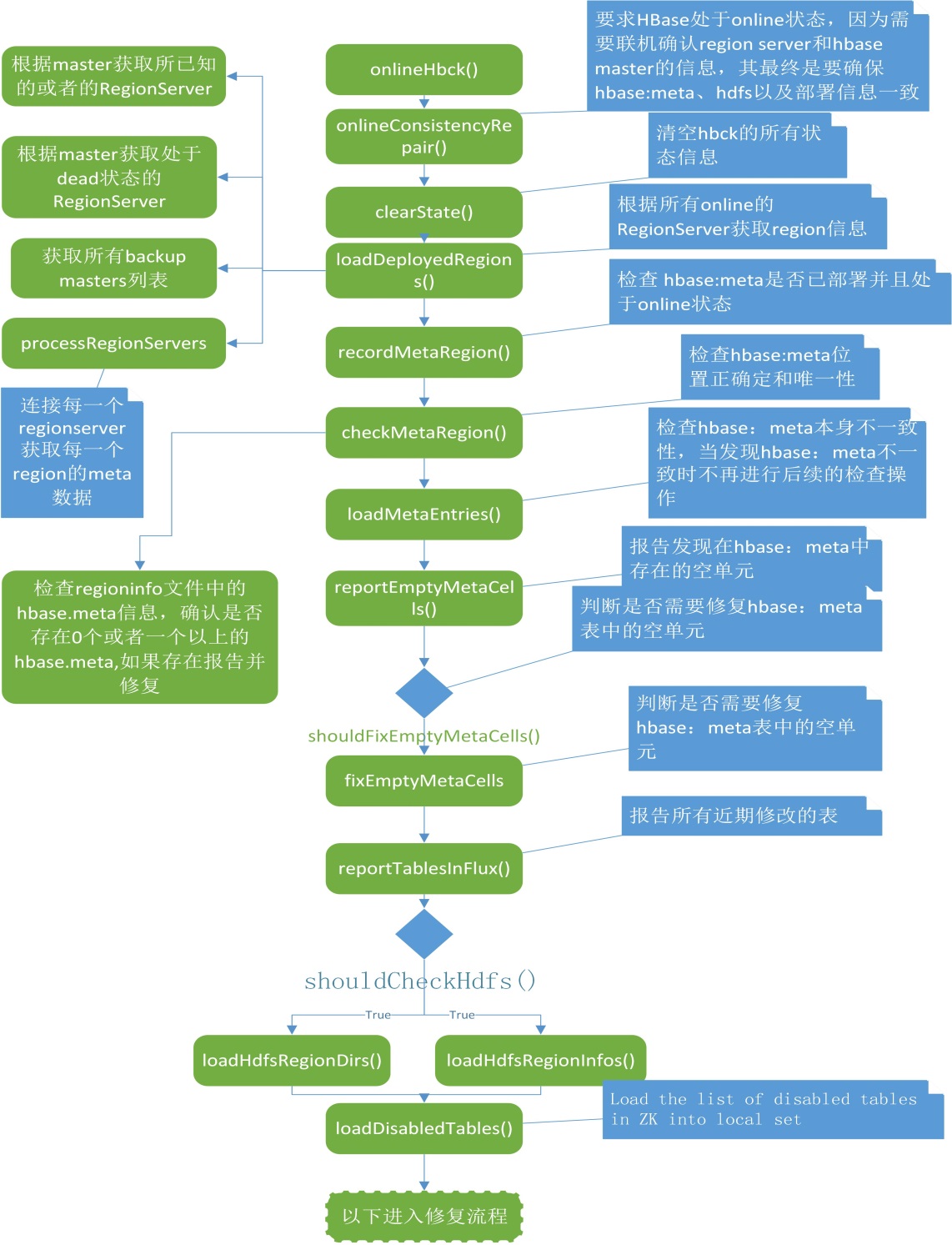
如图所示，所有问题的检查和修复操作的入口为onlineHbck函数，其定义了表完整性和分区一致性的检查和修复操作，offlineHdfsIntegrityRepair定义了表完整性分析和修复的操作流程，其中检查流程如下图所示：



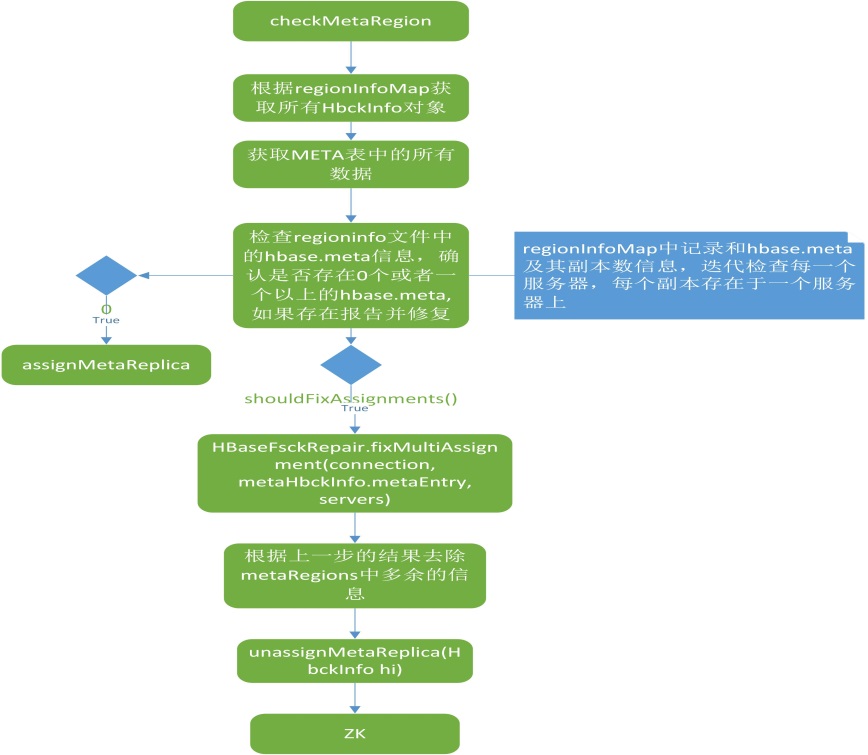
如图所示，该流程主要是校验表的完整性并根据参数进行修复，其主要扫描的表是regioninfo和tableinfo的数据情况。首先还是从获取conf信息开始，确定需要进行迭代修复的次数（hbase.hbck.integrityrepair.iterations.max 参数决定），为了保证此次hbck操作的有效性需要清除hbck的所有状态信息，然后调用restoreHdfsIntegrity对hdfs进行完整性检查和修复（根据传递的-fix参数）主要修复孤立分区、重叠分区等问题。因此其首先需要获取到所有分区的信息，restoreHdfsIntegrity会调用loadHdfsRegionDirs()函数扫描所有region并将扫描的结果映射到一个regionInfoMap，期间会对HBase的版本信息进行校验；之后会调用loadHdfsRegionInfos根据系统文件中加载来的regionInfos信息（已封装为regionInfoMap）填充HbckInfo，返回tablesInfo集合，tablesInfo是Tablename - > TableInfo的唯一映射，它包含了检查表一致性问题（漏洞，重复，重叠）所必需的结构化信息。如果tablesIncluded为空，则该映射包含所有表，否则，它只包含tableIncluded中的元表和表，除非指定了checkMetaOnly，在这种情况下，它只包含元表其中有一个很重要的操作loadTableInfosForTablesWithNoRegion是对虽然存在于hdfs中的表但是没有任何region报告其信息的的表的处理；接下来是根据hdfs的数据和加载的regionInfos信息进行hdfs完整性检查调用checkHdfsIntegrity，在checkHdfsIntegrity首先会根据hbasefsck.overlap.merge.parallel配置参数决定是否对重复分区的合并操作进行记录，并最终由TableIntegrityErrorHandler的具体实现对表完整性进行修复，TableIntegrityErrorHandler提供了回调函数来处理特定表的完整性问题，具体实现将在修复章节介绍，此处简单说明如果参数涉及fixHoles或者fixOverlaps的话会调用HDFSIntegrityFixer实现，其他完整性问题调用IntegrityFixSuggester实现，最后进行region链的检查，从Hbase:meta.表到region的完整链路信息。

1. **region一致性检查**

对分区一致性的检查要求HBase处于online状态，因为需要联机确认region server和hbase master的信息，其最终是要确保hbase:meta、hdfs以及部署信息一致，检查流程如下图所示:



进入onlineConsistencyRepair执行的第一个操作就是清空hbck的所有状态信息，这是每一个hbck操作必须进行的，防止历史状态影响hbck的执行；之后调用loadDeployedRegions根据online的regionserver获取所有region信息，在loadDeployedRegions首先根据master获取已知的处于活着状态的regionserver，然后是处于dead状态的regionserver以及backup masters节点信息，之后连接到每一个regionserver获取每一个region的metadata信息；之后调用recordMetaRegion记录在ZK中找到的hbase:meta的位置信息，其目的是检查hbase:meta是否已部署并处于online状态；然后调用checkMetaRegion检查hbase:meta位置的正确性和唯一性，其流程图如下所示：



其中对于未分配的hbase:meta会调用HBaseFsckRepair.fixUnassigned(admin, h)，HBaseFsckRepair.waitUntilAssigned(admin, h)，通过创建或者转换一个zk节点并分配给该region，然后使用一个特殊的标记标注此操作并告知master该操作是hbck的一个强制操作来修复此种情况，需要强制ZK下线。对于多于1个的情况则调用HBaseFsckRepair.fixMultiAssignment(connection, metaHbckInfo.metaEntry, servers)操作进行修复，通过强制下线多于的连接修复。经过以上的检查和修复操作基本保证hbase:meta的正确定，然后调用loadMetaEntries()，扫描所有hbase:meta，将找到的所有region映射到regionInfoMap，期间会根据RegionLocations.getRegionLocation(HRegionInfo.

DEFAULT\_REPLICA\_ID)、RegionLocations.getRegionLocation(HRegionInfo.

DEFAULT\_REPLICA\_ID).getRegionInfo()是否为null来判断该cell是否为空，并填充一个Set<Result> emptyRegionInfoQualifiers作进一步处理，通过reportEmptyMetaCells上报所有空的meta cell，通过fixEmptyMetaCells修复空的meta cell，最简单的操作是直接把该记录从hbase：meta中删除，之后调用reportTablesInFlux上报之前操作中修改过的所有的表信息，之后进去修复流程，首先会根据出入的参数决定是否进行HFiles的检查和修复，该流程已在之前的操作中介绍过，此处不再赘述。

###### 三 修复

首先需要明确的是不管是region一致性还是表的完整性问题，究其原因都是regioninfos和.META.之间的信息不完整或者不一致，因此修复过程也是对regioninfo文件和.META.表的操作。其实在检查的过程中已进行了一些修复操作，比如说：region一致性中对hbase:meta表的修复，在表一致性检查中多“孤立”分区、重复分区的修复，可以说检查和修复是一个混合的过程，每做一些关键的修复操作都会进行新的检查，然后一旦发现问题在进行修复。

对于表的完整性修复首先将扫描表所在目录下的.regioninfo文件，然后验证每个表的完整性，如果存在孤立的Region(没有.regioninfo文件的Region)或者Region Holes，将会创建出新的Region，填补上了这个区间，然后加上-fixAssignments  -fixMeta 来解决问题，(-fixAssignments 用于assign region），（ -fixMeta用于在META表中添加region的记录）。Backwards Region和degenerate Regions (即：endkey=startkey) 首先会将数据移到一个“sideline”目录中，然后在恢复数据，如果有重叠区域，则会创建一个新区域，并将所有数据合并到新区域。表完整性修复仅适用于HDFS且可以脱机处理，HBase Reginserver和HBase Master不需要运行，并最终以离线方式完全重建hbase:meta表。

对于区域一致性的修复需要三个条件 1），存在于HDFS Region目录中的.regioninfo文件是有效；2），META中的.regioninfo数据的有效性；以及3）Region仅被部署在Master上具有适合状态的Regionserver上。 region一致性修复要求hbase online，以便hbck可以连接到HBase Master和regionserver。执行hbck命令之前必须成功调用connect()函数，修复Region一致性问题的风险相对较小。

以下是一些修复命令，需要注意的时hbck支持命令合并，也就是数一条命令代表多条命令执行比如：/hbase hbck -fixAssignments -fixMeta –fixHdfsHoles和/hbase hbck -repairHoles

具有一样的语义，会执行同样的操作。

1，本地修复(region不一致性修复)

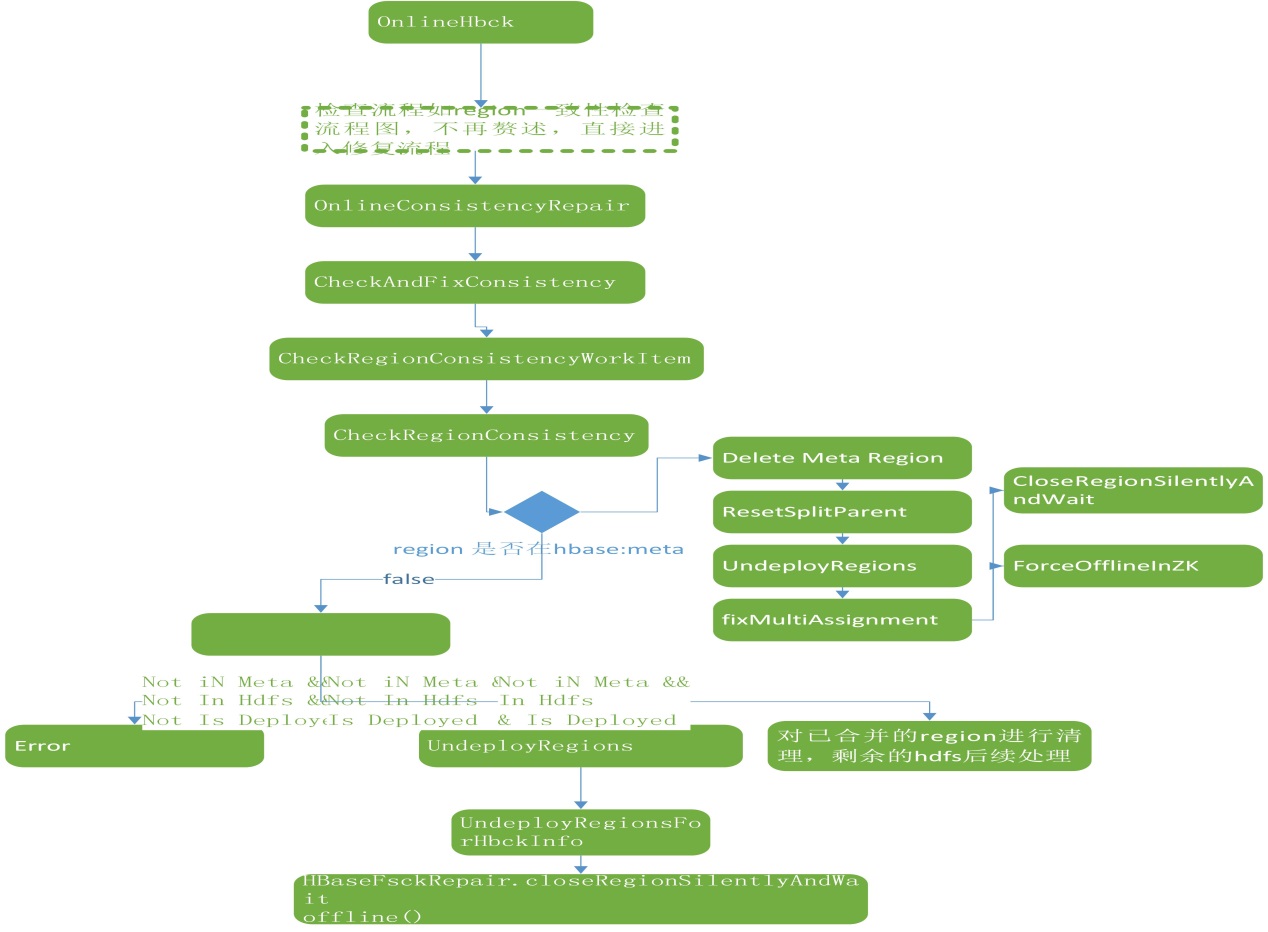
在HBase的修复过程应该严格基于最低风险原则，通常来讲应该先修复region的不一致性问题，因为该阶段仅需修改内存中的数据，暂时的zookeeper数据或META表中的黑洞。region一致性要求HBase实例具有HDFS（.regioninfo文件）中区域数据的状态信息，并且region row在.META表中。

可通过以下命令进行修复:

*$ ./bin/hbase hbck -fixAssignments*

用于修复region没有assign、不应该assign或者assign了多次的问题

其修复流程如下图所示：



*$./bin/hbase hbck -fixMeta*

如果在hbase:meta表中找到相关的region信息而在hdfs里没有找到，则移除这些行。如果这些region在hdfs里，则在hbase:meta里增加新的行，过程很简单不再赘述。

基于最低修复风险原则，通常来说以下三种表完整性问题的修复可以优先修复：

1，degenerate (startkey == endkey) regions；

2， backwards regions (startkey > endkey)；

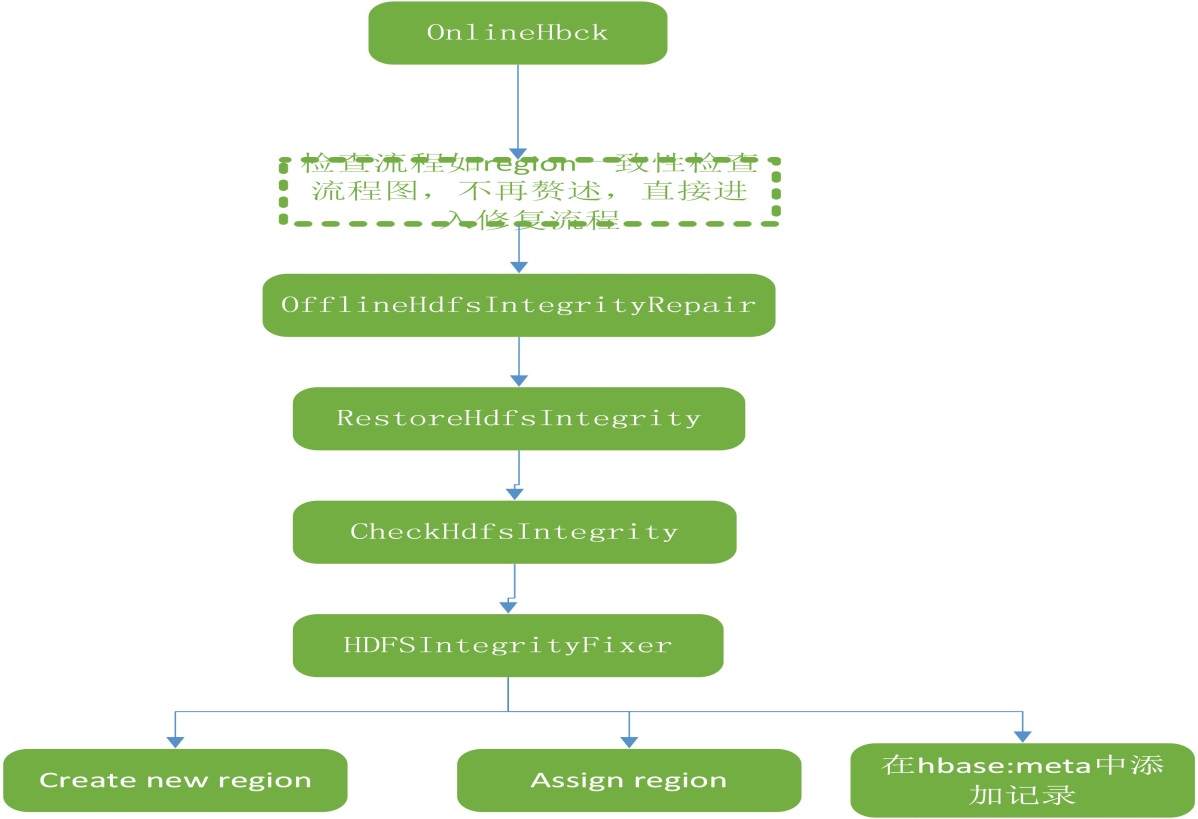
这两种问题的修复比较简单，可通过临时目录文件的复制、删除操作修复。

3，region holes

可以使用-fixHdfsHoles，在文件系统上创建新的空region修复,并增加-fixMeta和-fixAssignments选项以使新区域保持一致。）修复region holes时，-fixHdfsHoles 选项只是创建了一个新的空region，填补上了这个区间，还需要加上-fixAssignments -fixMeta 来解决问题，（ -fixAssignments 用于新region的assign），（ -fixMeta用于在META表中添加region的记录）。

*$ ./bin/hbase hbck -fixAssignments -fixMeta -fixHdfsHoles*

-fixHdfsHoles的执行流程如下：



由于该组合是一个常见操作可以使用以下命令代替:

*$ ./bin/hbase hbck –repairHoles*

 -repairHoles 修复region holes，相当-fixAssignments  -fixMeta  -fixHdfsHoles  -fixHdfsOrphans

2，重叠区域修复

表完整性问题的修复极大可能要操作重叠区域，这是一个风险较高的操作，因为它需要修改文件系统，并根据系统参数做一些决策，并且可能需要一些手动步骤。此时，最好先分析hbck -details运行的输出，并做好保护措施来限制修复的范围。-fixHdfsOrphans修复那些缺失region元数据的文件如(.regioninfo)的region目录；-fixHdfsOverlaps修复重叠的regions。

当修复重叠区域时，一个Region上的数据可以通过两种方式在文件系统上修改：

1）将区域合并到更大的区域；

2）先将数据移动到 “sideline”目录然后再恢复，通过迁移数据、恢复编译、WAL以及其他操作，至其中牵扯几个参数：

-maxMerge <n> （n默认是5），当region有重叠是，需要合并region，一次合并的region数最大不超过这个值。

-sidelineBigOverlaps ，当修复region overlaps问题时，允许重叠的最大分区数（修复后，可以把没有参与的数据通过bulk load加载到相应的region）

-maxOverlapsToSideline <n> （n默认是2），当修复region overlaps问题时，一组里最多允许多少个region不参与 修复。

-repair，相当于-fixAssignments  -fixMeta  -fixHdfsHoles -fixHdfsOrphans

 -fixHdfsOverlaps -fixVersionFile -sidelineBigOverlaps

此修复策略可分为两个步骤

1，修复HDFS上的表完整性(通过合并分区或者构造新的分区实现);

2，修复Region一致性。

###### 四 总结

关于HBCK工具核心思想在于hbase:meta、hdfs、ZK和region在region server上的信息的一致性，通过扫面hbase:meta、hdfs root目录（regioninfos）以及过程量如WorkItemRegion等收集和比较信息，发现问题，然后依然是操作这几个文件进行修复，期间需要借助“锁”来达到操作的互斥性。