**BRFS文件系统详细设计**

**Version 1.0.0**

**北京博睿宏远数据科技股份有限公司**

**变理记录：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **版本号** | **修改时间** | **修改内容** | **修改人** |
| 1 | 1.0.0 | 2018-1-2 | 文档初始化 | 魏征,陈俞朋,朱成岗,张念礼 |

# 引言

## 目的

本文档为文件系统服务端的详细设计说明书，详细说明了各个模块和各个层次的具体实现细节，为项目的需求分析说明书提供了完整的技术解决设计方案。同时，对软件应具有的功能和性能及其他有效性需求也提供了有效的实现方法。

该文档中主要解决实现该系统功能的各个模块的细节设计问题，以及模块与模块之间的交互问题的设计，为系统的编写提供了详细的指南，是进行代码编写的主要依据，也为系统的测试提供了方向。

本文档的预期读者有：本项目的审核者和平台的最终使用者，及系统维护人员、测试人员和数据库的维护人员等。

## 术语定义

* ***StorageName***：存储空间名称，简称SN，用户在存储数据的时候，必须指定一个StorageName，针对SN可以设置副本数、TTL等属性。
* ***ServerID***：服务标识，简称SID，每个服务在启动的时候，必须拥有一组服务标识。ServerID分为Single ServerID、Multi ServerID、Virtual ServerID。Single ServerID是针对单副本SN来使用，该ServerID不会过期；Multi ServerID是针对多副本SN来使用，该ServerID在副本恢复完成后会过期，并重新生成；Virtual ServerID是针对当副本数多于可用Server时，来完成虚拟存储时提供的Virtual ServerID。
* ***FID***：文件标识，由一系列编码过的字符串组成，用户需要提供相应的FID来读取数据。

# 系统结构



图-1 系统结构

# 模块设计

## 公共基础模块

### 配置文件

配置文件采用key-value的形式的properties文件。默认的文件名为server\_default.properties，用户自定义的文件名为server.properties。在自定义文件中，可以覆盖默认文件的参数，默认文件不提供修改属性。详情见具体配置文件config/server\_default.properties和config/server.properties。

### Zookeeper公共节点设计

Zookeeper节点设计规则如下：

1. 具有相同的clusterName属于同一个集群；
2. 集群可创建多个Storage Name；
3. 每个server注册临时节点，来表示自己加入
4. 整个集群具有自己的task存储和管理

Cluster之间的关系如下：每个Cluster中的节点设计如下：serverid sequence：维护自增ServerID的生成节点。

Servers：服务注册节点。

Locks：创建锁的节点。

storageNames：存储SN的节点。

Tasks：创建任务的节点。

DataRouteInfo：存储消息路由规则的节点。

UserInfos：存储用户信息的节点。

### 消息PB格式设计

网络通信采用PB序列化的数据，主要包括文件数据传输和文件存储。内部结构详情见FS\_Proto/proto/目录中的.proto文件。

## 客户端与服务端交互模块

### 创建StorageName

#### 程序描述

用户数据必须与一个StorageName进行关联，所以存数据时如果还没有StorageName必须先对其进行创建。

#### 具体功能

客户端提供接口使用户能与服务端进行通讯，以完成创建StorageName的目的。

#### 输入项

（1）StorageName名称

（2）Map属性

属性包括：

1、StorageName描述

2、StorageName存储数据的副本数

3、StorageName存储数据的TTL

4、StorageName是否归并

5、StorageName的文件大小上限

#### 输出项

创建StorageName的状态信息

成功 2000

已存在 4001

创建失败 4002

#### 时序图



1、当StorageName存在时，直接向用户返回已存在的响应；

2、如果StorageName不存在，则向StorageName仓库插入新StorageName的信息，新信息最终会以ZK节点的形式存储到ZK上。创建成功则返回成功响应；否则，返回错误码。

#### 测试计划

### 修改StorageName

#### 程序描述

修改StorageName对应的属性信息。

#### 具体功能

目前只提供修改StorageName的TTL属性的功能。

#### 输入项

（1）StorageName名称

（2）Map属性

目前提供修改的属性：

1、StorageName的TTL设置值

2、StorageName是否归并

3、StorageName的文件大小上限

#### 输出项

修改属性的结果状态信息：

成功 2000

不存在 4003

更新失败 4004

#### 时序图



1、更新StorageName信息前应该先检查其是否存在；

2、如果不存在，则返回不存在的错误响应；

3、如果存在，则更新StorageName的信息，最终写入到ZK节点中；

4、如果写入zk失败，则返回错误信息。

#### 测试计划

### 删除StorageName

#### 程序描述

当StorageName中存储的数据已经被删除，并且此StorageName不会再存储任何信息时，可以对此StorageName进行删除操作。

#### 具体功能

Client端向Server端发送StorageName删除请求，Server先检测StorageName是否还关联着用户数据，如果关联，则返回错误状态；否则，删除StorageName的所有元数据信息，并返回删除成功状态。

#### 输入项

（1）StorageName名称

#### 输出项

删除操作的结果状态信息

成功 2000

删除失败 4005

#### 时序图



1、如果StorageName不存在，则返回成功；

2、如果StorageName存在，且没有关联任何数据，则可以进行删除，如果删除成功则返回成功响应；

3、如果StorageName存在，但其有数据关联，则无法删除其信息，返回错误响应。

#### 测试计划

### 数据写入

#### 程序描述

用户通过调用Client端提供的接口向BRFS写入自己的数据，每次写入的数据可以保存多个副本，副本数由数据所在的StorageName指定。每条数据写入成功后写入模块都会返回一个对应的FID，用户后续可以通过此FID查询到这个写入的信息。

#### 具体功能

数据写入主要分3个部分：

（1）Client端写入接口

（2）Server端副本管理模块

（3）Server端磁盘管理模块

1、Client端接口提供给用户写入数据的入口，Client端收到用户的数据后，通过资源管理模块选择一个合适的Server，接着通过Http协议向此Server发送写入请求；

2、Server端副本管理模块中管理着文件列表，每个文件对应一个StorageName，关系如下：

StorageName：File1, File2, File3…

当副本管理模块接收到数据后，它会根据数据所属的StorageName把数据和某个文件关联，然后把（FileName, Data）的关联信息发送至磁盘管理模块。因为StorageName可能有多个副本，所以StorageName对应的每个文件会被多个磁盘管理模块记录，所以之前的（FileName, Data）数据会同时发送到多个磁盘管理模块，这也是数据多副本的实现方式；

3、磁盘管理模块接收到（FileName, Data）信息后，就是简单的打开FileName指定的文件，然后写入data，并把data在文件中的偏移量返回给副本管理模块。副本管理模块接收到data对于的偏移量信息后就可以生成数据对应的FID。

#### 输入项

（1）StorageName信息

（2）用户数据

#### 输出项

数据对应的文件名（FID）

异常结果：

StorageName不存在 5001

用户数据异常 5002

#### 时序图

（1）Client端数据写入时序图



1、客户端写数据前需要打开一个StorageName；

2、确定StorageName后，根据资源管理模块提供的信息选择一个合适的Server，然后向选定的Server发送数据；

3、数据保存成功后会收到Server端返回的FID。

（2）Server端副本管理模块写入时序图



1、Server端负责接收数据的模块是副本管理模块，收到数据后，先检测有没有可写入的文件节点，如果没有则创建文件节点，文件节点的创建过程包括文件名的创建和文件副本所在Server的选取。

2、文件节点准备就绪后，数据会被分配到所有副本Server上进行写磁盘操作。如果写入过程中发生了异常，比如某个写入服务挂了，或者写入服务的返回值不一致，这时应该把文件标记为异常状态，并将其交给同步模块进行数据同步，同步完毕后文件可以继续进行写入。

（3）Server端磁盘管理模块写入时序图



1、磁盘管理模块写数据前会先检测文件是否存在，如果不存在则新建文件；

2、数据在写入文件前会先生成CRC校验码，然后写到缓存中并返回数据在文件中的偏移量；

3、缓存中的数据会定期的同步到磁盘中。

#### 测试计划

### 数据读取

#### 程序描述

读取用户通过数据写入流程写入的数据。

#### 具体功能

FID中记录了数据所在的Server列表、磁盘的文件名以及在文件中的偏移量和大小，根据这些信息，磁盘管理模块可以直接进行磁盘读取获取数据。

Client端获取FID后解析出Server列表信息，然后向合适的Server发送读取请求，Server端的磁盘管理模块会根据文件名、偏移量及大小信息读取用户数据并返回给Client端。

#### 输入项

文件名（FID）

#### 输出项

用户数据

异常情况：

文件不存在 5003

文件内容异常 5004

#### 时序图

（1）Client端数据读取时序图



1、用户读取数据前通过打开操作选定一个StorageName；

2、然后根据FID中解析出的Server信息选择一个合适的Server；

3、向Server发送读取请求。

（2）Server端数据读取时序图



磁盘管理模块获取到数据所在的文件名、偏移量及大小后就可以直接从文件中读取数据，完成后进行CRC校验，如果匹配则返回数据；否则返回错误码。

#### 测试计划

### 数据删除

#### 程序描述

对某一时间段内的数据进行手动删除。

#### 具体功能

用户可以通过传输数据删除请求到Server端，请求中必须带有StorageName和时间区间两个参数，Server接收到删除请求后会向任务管理模块创建删除任务，然后向用户返回任务是否创建成功的响应。

#### 输入项

（1）StorageName

（2）时间区间

#### 输出项

数据删除任务创建的状态信息

异常情况：

任务创建异常 6001

#### 时序图

（1）Client端数据删除时序图



1、删除数据前需要通过打开操作选定一个StorageName；

2、随机选择一个可用Server后向此Server发送数据删除请求。

（2）Server端提交数据删除任务



（3）Server端磁盘管理模块数据删除时序图



数据删除任务的执行需要通过磁盘管理模块进行数据的删除。

#### 测试计划

### 异常情况处理

1、Client端如果连不上Server则会重复进行建连，特定次数后还没连上则抛出异常。

2、StorageName创建过程中，如果Server端内部创建失败则会视情况进行重试，例如向ZK中创建节点失败则对其进行特定重试次数，如果依旧失败则向客户端发送错误回应。

3、StorageName更新及删除的过程中如果在Server端内部操作失败也会进行重试。如果删除过程中StorageName内还有数据则直接返回失败。

4、数据操作过程前如果获取StorageName失败，则返回的StorageNameStick对象为空。

5、用户数据写入过程中，数据传到Server内部后，如果Server挂了，则Client端会等待响应超时，可以选取其他可用Server之后对数据进行重发。但数据成功写入所有可用的副本文件后向客户端返回数据对应的FID。如果一个文件的副本Server挂了，则文件可以继续写入，写入过程中Server恢复则会对写入文件进行数据同步，同步完毕后再进行写入；如果Server恢复时文件已经写入完毕则把文件标记为副本不足，后续对其进行恢复，而恢复的方式会依据副本恢复的时间而定，如果还没触发副本恢复则在Server启动时对文件进行同步；如果触发了副本恢复则直接删除文件。

6、数据的读取如果没法解析到可用的Server，则返回用户错误码。如果Server端没有FID中指定的文件，则返回客户端错误码，客户端检测到文件不存在的错误后会选择其他可用的Server进行重新读取，直到所有可用Server都尝试完。如果读取过程中出现数据校验无效，也会返回客户端错误码，并对文件进行标记以备后续恢复，客户端收到错误回应后会对其他可用Server进行重新读取。

7、数据删除是通过任务操作，如果任务创建失败，则返回客户端错误码，客户端会依次对所有Server进行重试，直到有一个成功或则全部失败。

## 数据检验模块

### 校验详情

1、数据写入检验

数据写入文件的同时会记录数据的元数据信息到日志文件中，元数据包括数据在文件中的offset、数据的字节数以及数据的CRC校验码。这些信息可以在文件写入中断时进行有效数据的检测。

服务启动时，副本管理模块会自动对正在写入的文件做校验，判断其数据的有效性并对其进行同步。

2、已完成文件内容检验

完整的文件内容校验用于副本迁移时文件文件内容的校验、文件校验任务的数据校验以及归并任务执行过程中数据有效性的检测。

（1）针对单条数据的CRC校验码可以根据用户的设置选择生成或不生成，如果生成校验码，则将其和数据一起保存到文件中；

（2）针对整个大文件的CRC检验码总是会生成的，并将其保存到文件的末尾。文件后续的迁移后一致性校验和数据定时校验等过程都会依据此校验码对文件的一致性和完成性做判断。

3、用户数据校验

用户数据从Client端到Server端，以及Server端的副本管理模块到磁盘管理模块之间的传输都可以根据用户的配置对用户数据生成CRC校验码，以便在传输过程中检测文件的有效性。

### 异常处理

1、写入文件过程中出现异常时，文件会先被移出可写文件列表，然后被提交到文件同步模块，同步模块会检测出所有文件的有效数据长度，然后以长度最大的文件对其他文件进行数据同步，同步完成后文件可以被继续写入。同步模块进行同步的时机是当前写入的文件出现不一致时，这个情况有两种：

（1）当所有副本Server都可用时，写入数据得到的响应不一致；

（2）当一个副本Server崩溃并在文件写入完成之前恢复。

2、文件内容校验失败后，异常文件的文件名会被提交到文件同步模块，同步模块会根据文件名中记录的Server列表选择其中一个Server进行文件修复。

## 副本管理模块

### 数据接入子模块

作为客户端数据的输出端，此模块负责接收来自客户端的数据，并且维护三种映射关系：

①StorageName到文件名之间的映射，即

（StorageName, [FileName1, FileName2, …]）

②用户数据到文件之间的映射，即

（UserData, FileName）

③文件到主机之间的映射，即

（FileName, [Server1, Server2, …]）

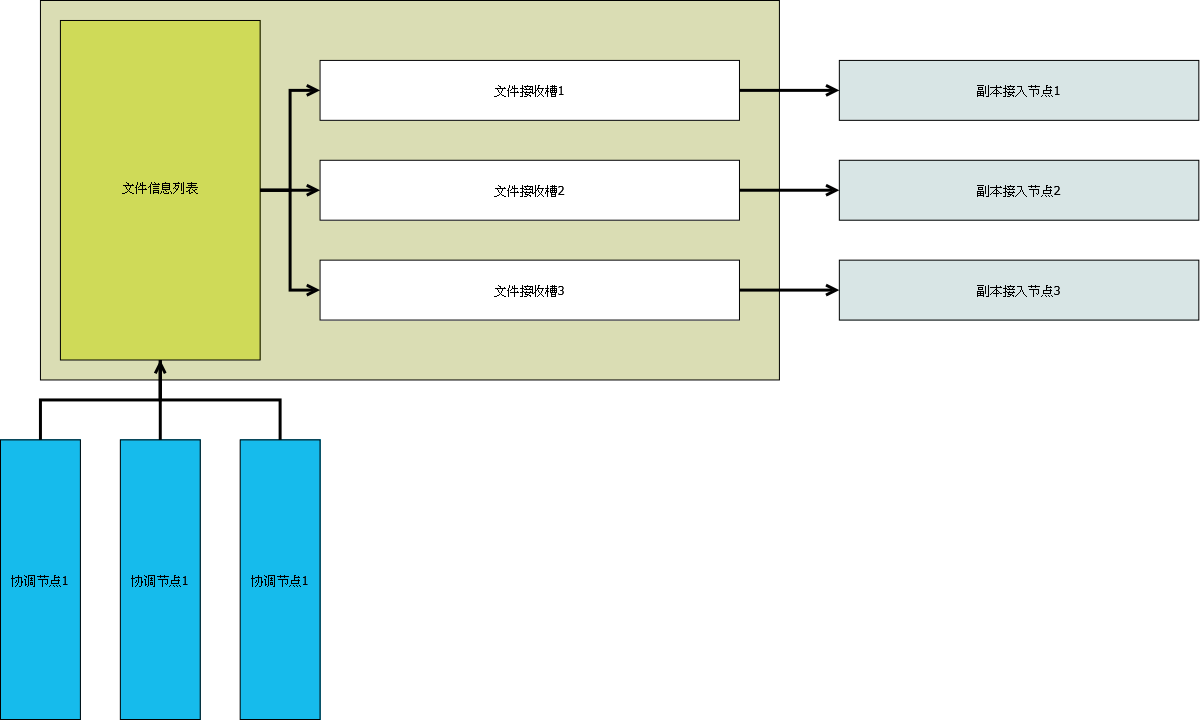
每个StorageName会有多个当前正在写入的文件，每条属于某个StorageName的数据都会被写入到这个StorageName对应的文件列表中的一个。当没有文件可以满足时，可以创建新的文件，文件创建过程中需要先根据StorageName设置的副本数选取相应数量的Server进行绑定（上述的关系③），然后向文件协调模块注册一个文件节点，在文件满足关闭条件被正常关闭之前，这个文件节点会一直存在。注册成功后这个文件才可以进行正常写入。

文件的关闭条件是大小达到某个设定的阈值。

### 文件协调子模块

这个模块的存在是为了处理由于系统崩溃导致文件不完整的问题。数据接入子模块中创建文件时要注册文件节点的目的就在此。当文件协调子模块检测到某个Server不能正常工作时，它会扫描出所有由异常Server管理的文件节点，然后把这些文件分配给其他还在正常工作的Server，这些文件就能继续进行写入了。

工作原理如下图所示：



1、数据接入子模块负责创建文件，并向文件协调模块发布文件信息；

2、文件协调模块收到数据接入模块传输的文件信息后会把其信息放到zookeeper中，其区域为图中的文件信息列表；

3、当一个文件正常写满并关闭后，数据接入模块会向文件协调模块发送关闭请求，之后文件从文件信息列表删除；

4、当一个Server出现异常，协调节点会扫描文件信息列表，筛选出异常Server拥有的所有文件，然后根据特定策略将其分配至正常Server对应的文件接收槽（只有文件对应的副本Server至少有一个存活时才会对其进行分配）；

5、正常Server会监听到文件接收槽中的内容变化，从槽中取出文件，放到自己维护的写入列表中。

### 数据同步子模块

当系统崩溃导致部分文件的内容在多个副本之间出现不一致时，数据同步模块就负责对比各副本文件之间的数据内容，然后根据有效数据最多那个副本对其他副本进行数据同步，只有文件各副本之间的数据处于一致状态时，文件才能进行新数据的写入。

文件的有效数据检测是通过数据写入日志完成的，每个正在写的文件都会维护一个写入日志，记录当前写入数据在文件中的offset、size和数据的CRC校验码。扫描这个日志文件就可以知道文件的有效数据长度。

## 任务处理模块

### 简介

#### 任务设计说明：

1. 任务信息存放在zk，由zk保存任务信息及状态。
2. 任务节点是有状态的，分为初始，运行，完成，异常，挂起；
3. 任务节点下面是服务任务节点，同样具有任务状态；
4. 服务任务节点数等于活动状态的服务数；
5. 服务任务节点的状态由各个服务维护，而且每个服务只对相同服务标识的节点负责；
6. 任务节点的FINISH标识，由最后一个执行完成的服务更新任务状态；

#### 任务的种类及相关说明

任务按调用者的身份可以分为系统任务与用户任务。

系统任务是由系统定时触发的任务。任务触发的间隔频率，及触发的任务类型，可通过配置文件进行修改。

系统任务可以分为 删除任务，归并任务，副本校验任务，文件校验任务，副本恢复任务；

系统任务的创建均是由集群中的任务创建服务创建。

用户任务是由客户端或命令行触发的任务。

### 选举系统任务管理服务

#### 程序描述

从集群中选举出系统任务管理服务

#### 具体功能

一个集群只能有一个系统任务管理服务，在集群启动时，通过zookeeper选举出一个任务管理服务。被选举出来机器为系统任务管理服务，剩余服务作为备用服务。

当系统任务管理服务异常时，可从剩余服务中选举出新的系统任务管理服务。

#### 输入项

无

#### 输出项

无

#### 时序图



时序图说明：

1. 是选举系统任务管理服务的入口方法，该方法的返回值类型为Boolean；
2. 是获取分布式锁方法，返回值类型为Boolean；
3. 是初始化系统任务管理服务方法；

#### 测试计划

### 创建任务

#### 程序描述

主要功能是生成任务信息，并保存到zookeeper上。

#### 具体功能

任务创建分为两种情况，服务定时创建任务，用户创建任务。

服务定时创建是由任务管理服务周期性创建各类任务。

用户创建是由客户端或命令行触发的任务创建。

#### 输入项

任务信息

#### 输出项

1. 状态码
2. 可执行的任务信息

#### 时序图

任务创建时序图：



时序图说明：

1是任务创建触发为任务创建的入口，TaskContent存储着任务的类型，SN，开始结束时间；

2是获取SN信息。2首先从本机内存获取SN信息，若内存的SN信息为空，2从ZK上获取SN信息。

3是获取到的SN信息。

4是过滤与任务无关的SN信息，并返回任务相关的SN信息。

5，8是返回为给调用者的信息；5为异常情况，当SN信息为空时返回0；8为正常情况，返回影响到的目录数。

6是将任务信息转换为可被服务执行的任务信息；

7是将可执行的任务信息注册到zookeeper上。

服务定时创建任务：



时序图说明：

1是检查是否服务持有任务创建锁；若持有任务创建锁，则触发任务创建，否则不触发；

2是任务创建触发为任务创建入口，TaskContent存储着任务的类型，SN，开始结束时间；

3 是任务创建入口的返回值，当为0时表示创建失败，不为0时表示要操作的目录数；

用户创建任务



时序图说明：

当有用户任务请求时，才能触发该流程。

1是任务创建触发为任务创建入口，TaskContent存储着任务的类型，SN，开始结束时间；

2是任务创建入口的返回值，当为0时表示创建失败，不为0时表示要操作的目录数；

#### 测试计划

### 执行任务

#### 程序描述

主要功能是处理任务信息涉及到的数据。

#### 具体功能

执行任务主要工作是根据任务信息处理相关数据。

在执行任务时，每个类型的任务都对应独立的线程池。这样不同类型任务的执行线程不互相影响。

执行任务处理的最小单位是SN下面的一个目录。

每个服务都会保存每个类型任务的上次执行任务名称，该数据存储在内存中。服务据此选择下一个执行的任务。当服务重启时该值重置。

#### 输入项

任务类型

#### 输出项

1. 执行的任务名称
2. 任务执行结果

#### 时序图

任务执行整体时序图



时序图说明：

1是任务执行的入口。TASK\_TYPE 是任务类型，根据传入的不同任务类型，获取不同的任务并执行。

执行任务的入口可被系统和用户触发执行，传入的参数不同后续执行也不同。

2是向资源管理获取执行权限，3是2的返回值，类型为Boolean，当为true时才能执行后续操作。

4是1的返回值，返回类型为string，为执行的任务名称；

5是从zk中获取任务，6为5的返回类型；5在获取任务信息时，同时会在相应的任务下创建服务的任务节点。

7是向资源管理获取任务执行的策略，8为7的返回值，根据服务的负载状况及任务影响的数据范围及优先级，决定连续执行的频率；

9是向任务执行序列请求创建一个任务对象，10是9的返回值；

11是将创建的任务对象提交到任务的线程池；

12是返回提交任务的名称

13是将任务执行的结果注册到zk上，13操作是异步执行的；

执行任务时序图：

时序图说明

1是创建任务方法，由TaskFactory根据传入的参数创建不同类型的Task；

2是不同类型Task的创建方法；

3是1的返回值，返回任务对象；

4是任务对象的线程提交方法，将任务执行线程提交到TaskPool。

5 是任务执行方法，不同的任务，RunTask方法的实现不一样

6是将执行结果注册到zk上，标识当前服务任务任务完成或异常

7是检查自己是否为最后一个完成的，返回值为各个服务的任务节点的状态；若返回的状态中有未完成的，则线程结束；若全部完成，则调用9。

9是将任务状态更新为完成或异常。

#### 测试计划

### 定期检查任务状态完整性

#### 程序描述

任务管理服务器周期性检查处于RUNNING的任务节点服务状态。及时更新任务状态。

#### 具体功能

任务管理服务器周期性检查处于RUNNING的任务节点服务状态。修改存在异常服务的任务的任务状态，确保任务的生命周期的完整。

#### 输入项

无

#### 输出项

1. 操作的任务个数
2. 更新任务状态

#### 时序图



时序图说明

1是定时检查任务的入口，

2 获取当前可用serverId集合，3为返回结果

4 获取为RUNNING状态的任务，5为返回结果

6检查任务中serverId是否存活，若不存活 则更新任务的状态为异常，7为更新操作

8 为异常任务的个数。

#### 测试计划

### 启动服务检查任务完整性

#### 程序描述

主要功能是服务启动时，保证执行任务顺序的完整性。

#### 具体功能

服务启动会根据服务标识判断服务是否为第一次启动，若不是则进行服务重启任务完整性校验。

服务重启任务完整性校验核心是获取该服务上次执行的任务名称。根据名称将未执行完的任务，未执行的任务重新执行。

#### 输入项

服务标识SID

#### 输出项

待执行任务名称

#### 时序图



时序图说明：

1任务完整性校验入口方法，

2 获取zk上对应任务类型的任务信息集合

3 遍历任务信息；从任务队列的末尾开始，找到第一次出现该serverId的任务 并将任务名称返回给调用者。

#### 测试计划

### 系统删除任务

系统删除任务主要是将过期的数据删除。

系统删除任务默认是开启状态的。

系统删除任务的创建及执行均由系统定时触发。

当系统删除任务执行时，根据任务信息，删除对应的sn下面的目录。

任务的最小操作单位是sn的一个目录。

任务的创建及执行详见创建任务，执行任务

### 系统归并任务

系统归并任务是根据配置信息，将数据归并到指定的存储介质中。

系统归并任务默认是关闭的，需要用户手动配置才能开启。

系统归并任务的创建及执行均是系统定时触发的。

系统任务的最小操作单位是sn的一个目录。

系统归并任务会对传输过程中的数据做文件校验，该功能可通过配置文件配置，默认关闭。

任务的创建及执行详见创建任务，执行任务。

### 系统副本校验任务

系统副本校验任务是对集群中所有sn的文件副本数进行校对，将副本数异常的文件列表注册到zk。

系统副本校验任务默认是关闭的，需要用户手动配置才能开启。

任务的创建及执行由系统定时触发 或命令行触发。

任务的创建及执行详见创建任务，执行任务

### 系统文件校验任务

系统文件校验任务是对集群中所有sn的文件进行CRC校对，将异常的文件列表注册到zk。

系统文件校验任务默认是关闭的，需要用户手动配置才能开启。

校验任务的创建及执行由系统定时触发 或命令行触发。

任务的创建及执行详见创建任务，执行任务

### 系统副本恢复任务

将挂掉服务的多副本数据，按副本恢复规则复制到指定服务。

系统副本恢复任务执行时，其他任务将被挂起。

任务的创建及执行详见创建任务，执行任务

### 用户删除任务

用户删除任务主要是通过客户端或命令行将需要删除的数据进行删除。

任务的创建及执行详见创建任务，执行任务

### 内部接口

#### 任务查询

主要查询任务的执行状态及任务内容

Int queryTaskStat(String taskName);

TaskContent queryTaskContent(String taskName);

#### 异常处理

当任务执行异常时，对数据的处理操作

Void taskExceptionHandler(TaskContent taskContent);

#### 任务处理

任务正常处理流程

Int taskHandler(TaskContent taskContent)

#### 任务回滚

任务回滚操作

Int taskRollBackHandler(TaskContent taskContent)

#### 任务暂停

杀死任务进程，并记录当前任务的记录

Int breakTaskContent(TaskContent taskContent)

#### 任务冲突检查

检查冲突任务，冲突的任务名称集合

String[] checkTaskCross(TaskContent task, TaskType[] taskTypes )

### 异常列举情况

**当任务执行时，由于某种原因造成服务挂掉，导致具体服务任务节点的状态无法更新，一直保持RUNNING，导致任务状态无法结束；这时由定时任务检查来保证任务状态的完整性**

**情景一：**

假设集群有服务A，B，C；任务t1创建成功，t1节点下有相应的服务A，B，C 三个服务的任务节点；

在服务A 执行t1任务时，由于某种原因，导致服务A 挂掉并永久不可用，在服务B，C执行完任务t1时，由于所有的节点并没有finish或exception，导致任务t1的状态无法更新。

这时定时任务检查线程工作，当检测到任务t1下面服务A的节点状态为RUNNING并且服务A为不可用状态时，会将任务t1更新为EXCEPTION，以此确保任务的一致性。

**当由于任务挂掉而错过某个任务的执行，会通过服务重启的任务完整性校验来保证任务的完整性**

**情景二：**

假设有集群有服务A，B，C；在任务t1被执行完后，由于某种原因导致A服务挂掉，之后创建任务t2，由于服务A挂掉，导致任务t2节点下并没有服务A，

在任务t2执行完后，服务A被重启，A根据任务完整性规则，从最后一个正确执行的任务t1 之后检查任务完整性，当扫描到t2时，发现没有服务A的任务节点，服务A在t2任务下创建节点，并将任务状态更新为RUNNING；当服务A执行完任务t2后，由服务A将t2的状态更新为FINSH，这样可以保证任务被执行完成

**当集群异常时，发生副本自动恢复机制，其他任务将被暂停，任务状态更新为挂起，并将任务内容提供给副本恢复机制处理，待副本恢复机制完成后。在执行相应的任务。**

**情景三：**

假设有集群有服务A，B，C；在某一时刻，B节点出现异常；集群触发副本恢复机制，此时有执行的删除任务t1,归并任务t2，校验任务t3。

副本恢复机制会调用接口，将正在执行的任务t1,t2,t3暂停，然后执行副本恢复，待任务执行完成或异常后，继续执行任务t1,t2,t3

## 资源管理模块

### 简介

资源管理模块收集集群机器资源负载情况，并平衡机器写入读取压力，保证集群正常提供服务。

资源管理模块根据不同的场景，提供不同的处理规则。确保集群正常提供服务。

资源管理模块将场景分为集群场景与任务场景。

#### 集群场景

集群场景是集群级别的事务处理，需要获取到各个服务节点的资源情况。在排除异常服务后，根据算法获取最优的服务节点，或最优的服务节点集合。

#### 任务场景：

任务场景是单机级别的事务处理，只对单个服务节点的任务做处理。需要获取单机的资源情况，任务的类型及任务信息。根据算法判别一个任务的执行与否及执行的任务的节流情况。

### 采集资源信息

#### 程序描述

主要采集机器的负载信息。

#### 具体功能

采集磁盘信息有以下几个指标：

磁盘空间大小，剩余空间大小，磁盘的读写速度，磁盘的最大IO速度；

采集网络信息有以下几个指标：

网卡的读写速度，网卡的最大IO速度；

采集内存信息有以下几个指标：

内存空间大小，内存使用率

资源信息采集周期性采集资源信息，

#### 输入项

无

#### 输出项

1. 机器原始负载信息
2. 机器基本信息

3.资源管理信息

#### 时序图

采集服务器状态信息时序



时序图说明：

1. 为外部入口，用来执行采集任务。
2. 为采集服务原始信息，返回状态信息ServerStatVo
3. 将状态信息存储到队列中
4. 获取机器基本信息，返回服务基础信息ServerVo
5. 将服务基础信息缓存到内存
6. 将服务基础信息保存到zookeeper

二次处理时序



时序图说明：

1. 外部调用入口；
2. 获取获取原始状态队列的长度，当长度小于2时，不进行二次处理；
3. 获取队列所有的状态数据，返回List类型；
4. 获取本机服务基础信息，返回ServerVo；
5. 获取集群其他服务基础信息，返回List类型；
6. 二次处理原始状态数据，返回ResouceVo；
7. 将ResouceVo缓存到内存；
8. 将ResouceVo保存到zookeeper；

#### 测试计划

### 获取可用服务

#### 程序描述

获取可用服务属于集群场景，在操作集群时，不同的场景选取不同的算法，计算出适合的服务进行操作。

#### 具体功能

获取可用服务属于集群场景，在操作集群时，不同的场景选取不同的算法，计算出适合的服务进行操作。可以分以下几个场景：

数据写入场景是客户端向集群写入数据时，选取合适的服务提供写入服务。

数据读取场景是客户端向集群请求读取数据时，选取合适的服务提供读取服务。

数据操作场景是客户端向集群发送集群操作信息的。例如创建SN，修改SN，删除SN，删除数据。

#### 输入项

1. 场景分类
2. 异常的服务标识SID

#### 输出项

1. 服务信息
2. 服务信息集合

#### 时序图



时序图说明：

1. 获取可用服务，返回一个服务的状态信息
2. 获取可用服务，排除异常的服务，返回一个服务的状态信息；
3. 获取可用服务集合，返回可用服务集合；
4. 获取可用服务集合，排除异常的服务，返回可用服务集合；

#### 测试计划

### 可执行任务服务

#### 程序描述

主要判断当前服务状态是否可执行任务。

#### 具体功能

可执行任务服务属于任务场景，主要判断单机的负载高低，决定任务是否执行。

#### 输入项

1. 任务类型
2. 负载情况

#### 输出项

Boolean

#### 时序图



时序图说明：

1.判断当前负载情况是否可以执行某个任务。

#### 测试计划

### 任务执行策略选择服务

#### 程序描述

主要功能是根据任务内容及类型，来决定任务频率；

#### 具体功能

根据任务操作的数据量，决定任务执行的频率；

#### 输入项

任务信息

#### 输出项

任务执行策略

#### 时序图

时序图说明：

1.获取任务执行策略，返回执行策略；

#### 测试计划

### 内部接口

#### 资源管理可用服务接口说明

1.选取最可用server，排除异常的sever集合

public ServerVo selectAvailableServer(List<Integer> exceptionServers);

2.选取最可用的server

public ServerVo selectAvailableServer();

3.选取可用server集合，排除异常的server集合

public List<ServerVo> selectAvailableServers(List<Integer> exceptionServers);

4.选取可用的server集合

public List<ServerVo> selectAvailableServers();

#### 资源管理任务可执行接口说明

1.任务可执行判断接口，根据任务类型判断

public boolean taskRunnable(TASK\_TYPE taskType);

2.任务可执行判断接口，根据任务类型及服务器负载状态

public boolean taskRunnable(TASK\_TYPE taskType, ServerVO servervo);

3.任务执行策略获取

public RunPattern taskRunnPattern(TaskContent task);

4.任务执行权限自定义接口

TaskLevel defineTaskLevel(TaskLevel currentTaskLevel);

### 异常情况

**任务执行时如何确保服务的健壮，避免因任务执行占用过多资源，导致程序写读请求受到影响**

1. 任务执行前会判断当前负载是否超过阀值，若没有超过阀值时，任务才会提交。
2. 任务在执行中，会根据任务的类型及数据量，决定一次处理的数据多少。
3. 在负载超过阀值时，必须至少有一个的任务可以被提交执行

## 负载均衡模块

### 背景

BRFS为分布式文件系统，客户端在选择服务端进行服务的时候，尽可能最大程度的利用各个服务器的资源，这样才能让整个系统达到最大的可用性。因此，BRFS自带内部的服务器均衡策略。

### 原理

BRFS的中的客户端请求大致分为3类：

1. 轻负载请求；
2. 写负载请求；
3. 读负载请求。

轻负载请求包括客户端创建SN、修改SN、删除SN，异步删除文件；写负载请求主要是客户端写文件操作；读负载请求主要是客户端读文件操作。

每一种请求对资源的敏感性不同，轻负载请求只对元数据进行处理，请求操作少，消耗资源少；写负载请求对服务端的空间剩余最为敏感，其次是服务端的硬盘I/O使用率；读负载请求需要在多个服务标识中选择一个最优服务端的服务标识来进行服务，主要是对服务端的硬盘I/O使用率较为敏感。

所以，对于轻负载请求，我们使用随机策略来选择服务端；对于写请求负载，需要通过计算空间剩余率和当前I/O使用率，基于概率论来选择服务端；对于读请求负载，需要通过文件路由规则来确定真正的服务端，再通过当前的I/O使用率，基于概率论来选择服务端。

### 负载均衡接口实现

### 程序描述

对于分布式服务系统，负载均衡是不可缺少的。一般常见的使用Nginx做负载均衡服务器，但是对于BRFS，Nginx却不能胜任，所以，为了尽可能的让服务端的资源合理利用，我们有必要写出自己的负载均衡服务，该程序主要解决客户端在选择服务端时的负载均衡问题。

#### 具体功能

根绝负载均衡策略为客户端选择一个最优的服务端来进行服务。

#### 输入项

1. 收集到的资源对象
2. 对于读负载请求所需要的FID

#### 输出项

1. 封装的NettyServer对象

#### 主要方法设计

1. **获取一个可用的server**

NettyServer getAvailableServer(SelectType selectType)

参数：SelectType 以那种方式来选择server。（RANDOM\_LOAD，WRITE\_LOAD，READ\_LOAD）。

返回值：NettyServer 封装的该server的对象，可直接使用该对象来进行请求。

1. **以随机的方式获取一个可用的server**

NettyServer randomServer()

返回值：NettyServer 封装的该server的对象，可直接使用该对象来进行请求。

1. **以fid来计算特定的server**

NettyServer specificServer(String fid)

参数：fid为文件标识，需要解析fid来计算选取特定的server。

返回值：NettyServer 封装的该server的对象，可直接使用该对象来进行请求。

1. **根据负载来进行计算选取可用的server**

NettyServer calcLoadServer()

返回值：NettyServer 封装的该server的对象，可直接使用该对象来进行请求。

#### 测试计划

## 副本自动平衡模块

### 背景

在设计brfs时，为了能提高分布式的最大优势，故设计各个server之间是对等的关系。也就是每个server都是独立的，并且能够共同存储和读取数据。

因此提出了serverId的概念，即服务标识。serverId可以让我们知道那些数据存在那些server中，而返回给客户的fid中就具有serverId。客户端可以通过简单的解析fid，就能够快速的定位到相应的server中读取文件。

Brfs是支持副本数的，若副本数大于1，那么同样的副本，会存放在不同的server中的。

如果某个server挂掉，那么之前位于这个server中的数据，就会发生副本迁移。迁移之后，还得保证迁移的数据能够正常的读取。此时我们使用一种消息路由表算法来进行数据迁移和数据读取。

### 消息路由表算法原理

在讲路由表算法之前，我们可以先听一个生活中经常遇到的事。

假如某公司目前有A,B,C,D四个员工，这四个员工分别持有各种不同的消息。某一天，A员工要离职，那么公司会让A员工把自己所知的消息告诉其他人。因为A持有的消息可能比较多，所以只告诉其中的一个人，那么他的工作压力会变大。因此，公司会通过消息的特征，将A知道的消息尽可能的均匀的告诉其他的三个人。公司只需要记录接手A消息的人，即[A->B,C,D]。此时，若来了两个新员工E,F。过了不久，C也离职了。那么C同样会把消息尽可能的均匀的告诉其他人，公司同样需要记录接手C消息的人。即，[C->B,D,E,F]。

若某一天，客户来找A询问一个消息，但是A已经离职了，那么公司会通过[A->B,C,D]以及消息特征来寻找接手的人。若该消息命中B，因B还在，所以B可以为客户服务。若命中为C，因C也已离职，则继续通过[C->B,D,E,F]以及消息特征来寻找接手的人。若此时命中的是E，因E存在，则由E来为客户进行服务。

在上面的故事中，我们知道接手记录是非常重要的。因为每一次交接，都会记录一次交接记录，方便根据不同的消息特征，来分配不同的人来接手。后面在找由那一个人接手的具体任务时，也有据可循。

上文说的接手记录或交接记录就是我们关系的路由表。

假设有有序s1-s6个节点，有副本xxx\_s1\_s2。当s1挂掉，则由s2来协助副本恢复。迁移记录1[s1->s2,s3,s4,s5,s6]，由于s2已经保存副本，故排除s2，有[s1->s3,s4,s5,s6]，假设迁移到s3上，则s3有副本xxx\_s1\_s2。我们可以通过规则[s1->s2,s3,s4,s5,s6]将查询s1上xxx\_s1\_s2的副本，都可以导航到该s3节点上。假设又新启动了s7节点，此时s2节点挂掉。则有迁移记录2[s2->s3,s4,s5,s6,s7]。根据迁移记录1，我们可以知道s3上已经有了副本，则副本xxx\_s1\_s2只能迁移到[s4,s5,s6,s7]，假设迁移到s4上。则s3,s4上有副本xxx\_s1\_s2。查询s1上的该副本会根据迁移记录1导航到s3上，查询s2上的该副本会根据迁移记录2导航到s4上。此时，若s3挂掉，有迁移记录3[s3->s4,s5,s6,s7]根据迁移记录1，可知s3上有需要恢复的xxx\_s1\_s2副本，根据迁移记录2，可知由s4来恢复。则xxx\_s1\_s2可恢复在[s5,s6,s7]上，此时选择s5来恢复。则s4和s5上有副本xxx\_s1\_s2。查找s1上的该副本，可通过迁移记录1和迁移记录3来查找，查找s2上的该副本，可通过迁移记录2来查找。以此类推。

优点：整体来看，每次在发生故障迁移的时候就会记录一条迁移规则，也就是说发生1000次迁移，只会产生一千条迁移记录。最差需要计算计算1000次（该条数据每次迁移的机器都挂掉，发生概率非常低），最好的性能为计算1次，时间越靠前的查找次数越低。对于新入的正常数据，查询都是一次性命中。只记录迁移记录，节省了大量的空间来记录每份文件的迁移关系。而对于新入的数据，都是一次性命中。

### 异常重选过滤策略

如果副本自动平衡正在恢复中，此时若某一个服务出现异常，那么副本会中断副本恢复，等待片刻，若异常服务还是没能恢复，则撤销当前操作，重新进行路由策略选择。如，当[s1->s2,s3,s4,s5,s6]刚开始恢复或者恢复一半时，s2出现异常或者挂掉时，此时会暂停任务等待一定的时间，若等待时间过后，s2还是没用恢复，则进行路由策略重新选择，若此时选择[s1->s3,s4,s5,s6]，若s2后面又回来，则只需要按照[s1->s3,s4,s5,s6]策略进行文件迁移一次即可。

### 算法设计

为了确保副本能够正常恢复，算法在设计时，需要遵循以下规则：

1. 以SN进行副本迁移和恢复。BRFS是以SN来进行数据隔离的，各个SN之间的数据互不影响。在数据副本恢复和迁移的时候，各个SN也应该互不影响。客户端在进行访问Server之前，只需加载需要的SN的消息路由规则。
2. Virtual ServerID优先进行副本迁移。此时可用的BRFS的Server少于该SN的副本数，BRFS服务异常或宕机导致的副本迁移无法执行，所以应该确保Virtual ServerID优先进行副本迁移。Virtual ServerID和Multi ServerID是选择一对一还是一对多都可以。遇到有Virtual ServerID的副本，必须检测是否能转换成Multi ServerID。
3. 选择供体Server列表和受体Server列表。供体Server即提供副本来恢复丢失副本的Server，受体Server，即副本迁移恢复所选的新Server。正常情况下，供体Server列表和受体Server列表一致，在某些情况下，如某些Server的硬盘空间不足，则不会充当受体Server，此时供体Server列表和受体Server列表不一致，但是，受体Server列表必是供体Server列表的子集。
4. 恢复之前，检查SN的副本是否可恢复。对于多副本的SN来说，只要所宕机的Server至少还有一台，则可以进行副本恢复，否则无法进行恢复恢复，此时将该SN标为异常SN（副本缺失的SN）。
5. 恢复之中，若供体Server宕机，也将该SN标为异常SN。
6. 异常SN的副本恢复应该暂停。异常SN不能继续进行副本迁移和恢复，必须等服务稳定则可继续迁移。
7. 异常SN的消息路由规则可进行分裂。此处为了最大程度的保护副本数，若异常SN迟迟没有恢复执行，又因为之后的该SN的副本数是完整的，但是因为异常SN的问题，导致后面的Server宕机无法进行SN迁移恢复。此时可以对SN进行时间分段。通过SN和时间来选择消息路由规则。
8. 副本迁移恢复的时候，以时间目录为单位，以时间的正排序进行恢复。若当前目录的某个文件在使用中，如文件还未写满，则进行文件监听，并延后处理。
9. 副本迁移的时候需要记录副本迁移日志。
10. 只有当该SN的副本迁移恢复正常结束，才能进行该SN的消息路由规则发布。

### 主要方法接口设计

void notifyAndUpdate(List<ServerInfo> beforeList,List<ServerInfo> currentList);

说明：若有服务挂掉或加入，则可以进行调用，用以生产任务或恢复认为或暂停任务。

参数：

1. 之前的ServerInfo列表
2. 当前的ServerInfo列表

Void recover(ServerID serverID，List<ServerID> outputServerID List<ServerID> inputServerID)

说明：将VirtualServerID的副本迁移到可用的ServerID中。

参数：

1. 要进行迁移的ServerID，包virtual ServerID
2. 供体ServerID列表
3. 受体ServerID列表

List<ServerID> getInputServerIDs(List<ServerInfo> currentList)

说明：获取受体ServerID列表

参数：当前服务信息列表

返回值：受体ServerID列表

List<ServerID> getOutputServerIDs(List<ServerInfo> currentList)

说明：获取供体ServerID列表

参数：当前服务信息列表

返回值：供体ServerID列表

Void writeRcoverLog(String log)

说明：记录迁移日志

参数：日志信息

Void publishRouteRole(RouteRole role,ZKClient client);

说明：发布正常的迁移规则

参数：

1. 路由规则对象
2. Zk客户端

### 测试计划

# 外部接口设计

## 客户端接口：

（1）BRFileSystem



BRFileSystem是一个总接口，可以通过它操作StorageName。

（2）StorageNameStick



用户使用这个接口写入、读取和删除数据，这个对象不能自己创建，只能通过BRFileSystem的openStorageName函数获取。

# 系统安全设计

## 权限管理

### 介绍

BRFS作为一个文件系统，为了保证用户的资源的安全性，权限管理则是不可缺失的。BRFS提供的Client需要进行身份认证，才能进一步去访问Server的资源。

在BRFS中，主要有两个操作涉及到权限认证：

1. StorageName的创建、删除、修改操作；
2. 文件的读、写、删除操作。

在进行以上的操作的时候，会进行权限检查，只有符合权限的操作才能正常进行，否则拒绝服务。

BRFS权限管理的特点：

1. 在初次启动Server的时候（同一个集群，只需要初始化一次），需要进行root用户密码的初始化；
2. 默认root用户具有最高的权限；
3. 由root用户来创建其他用户并授权；
4. 由root用户来维护StorageName的对应关系。

### 数据格式

用户权限信息会采用JSON数据格式保存在Zookeeper中。具体数据格式如下：

{

“user\_name”:””, //用户名

“password”:””,(加密) //密码

“acl”:3,(1个byte) //权限

“storage\_names”:[] //可以操作的storageName

}

user\_name：用户名，用户名具有唯一性；

password：密码，需要进行加密；

acl：访问权限，由一个byte来保存，其中前5位可用，后3位暂时保留，前五位分别对应cuwrd，其中，c代表创建SN，u代表修改SN，w代表写文件，r代表读文件，d代表删除SN和文件，设置相应的权限，则该位标识1，否则标识0；

storage\_names：可以操作的SN列表。

### 初始化root用户接口实现

#### 程序描述

BRFS在初次使用的时候，需要一个root用户来进行权限认证和数据操作，该程序在Server初始化时，进行root用户初始化。

#### 具体功能

初始化root用户，设置用户密码。

#### 输入项

1. root用户密码，string类型。

#### 输出项

1. 执行结果，boolean类型。

#### 时序图



#### 测试计划

### 认证接口实现

#### 程序描述

每个客户端在进行访问服务器的时候，必须经过身份权限认证。认证成功后，会给客户端返回一个token，客户端保存该token，以后每次访问，只需检查token是否有效即可。

#### 具体功能

进行客户端的身份权限认证。

#### 输入项

1. 用户名，string类型
2. 用户密码，string类型

#### 输出项

1. token令牌，string类型，若为null则认证失败。

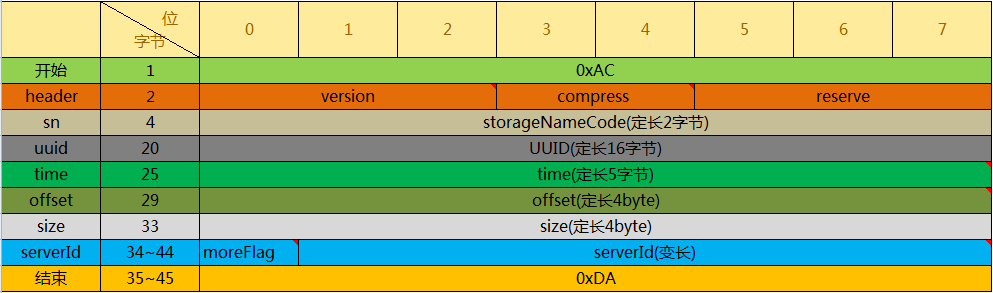
#### 时序图



#### 测试计划

## 数据传输

### FID编码



说明：

storageName 存储空间名称(也可有来区分不同的业务)

uuid 文件唯一标识

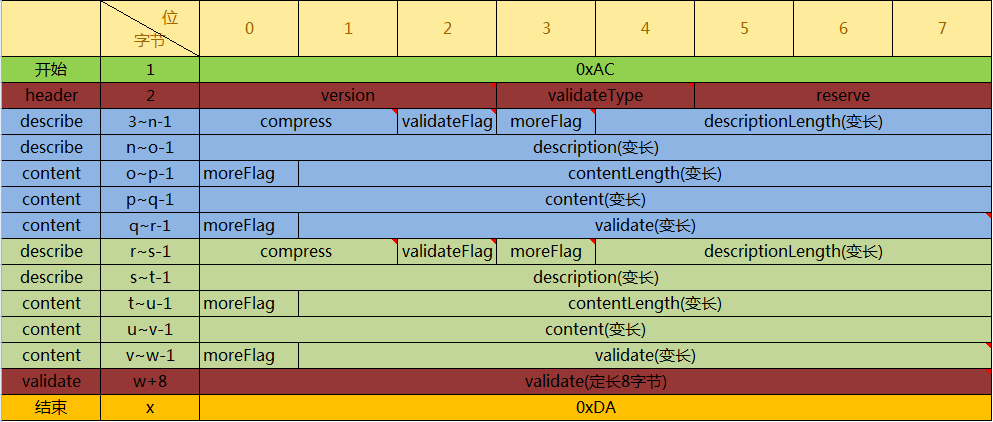
time 时间串(精确到分钟)

serverId 服务标识

offset 当前文件所在大文件里的偏移量

size 当前文件的长度

### FILE编码



说明：

文件内容编码：

上图编码中是一个大文件包含两个小文件

文件路径：

/brfs/storageName/yyyyMM/dd/hh/mm/uuid\_serverId

其中time为时间戳上的后五位(秒与毫秒)对应的数据值

对于time的说明:每天1亿文件,每个5k,共500G,每秒有6M左右,我们一个大文件64M,大概每10s生成一个文件.

## 错误码

为方便排查问题以及规范程序中的错误描述，特设计了相应的错误编码。

**成功：**

任何处理成功，都返回2000。

**不成功：**

3000~3999，认证异常预留；

4000~4999，StorageName异常预留；

5000~5999，数据异常预留；

6000~6999，任务异常预留；

7000~7999，资源异常预留；

8000~8999，检验异常预留。

**详见<<ReturnCode.proto>>**