SSM 简介

随着大数据技术的使用，硬件技术不停的在发展，Hadoop集群也在适应大数据的各种使用场景。在大数据集群中，可以使用最新的存储介质包括Optane Memory、Optane SSD、Nvme SSD等来提高存储性能，在软件层面Hadoop也提供了多种功能如HDFS Cache、异构存储及EC等技术来提高文件的访问效率及降低存储成本，在Hadoop中存储的需求包括：

1. 文件大小不同，既要支持大文件的高性能处理，也要考虑小文件的存储
2. 数据的冷热程度不同，根据访问情况，将其存储到不同的存储介质上，根据存储介质的特点，可以将存储分为以下几层：

* 内存（Memory），缓存最热的数据
* NVMe SSD，高速SSD，适合存储热数据
* SATA SSD，新HDD，适合存储暖数据
* HDD，普通磁盘，量大价廉，适合存储冷数据

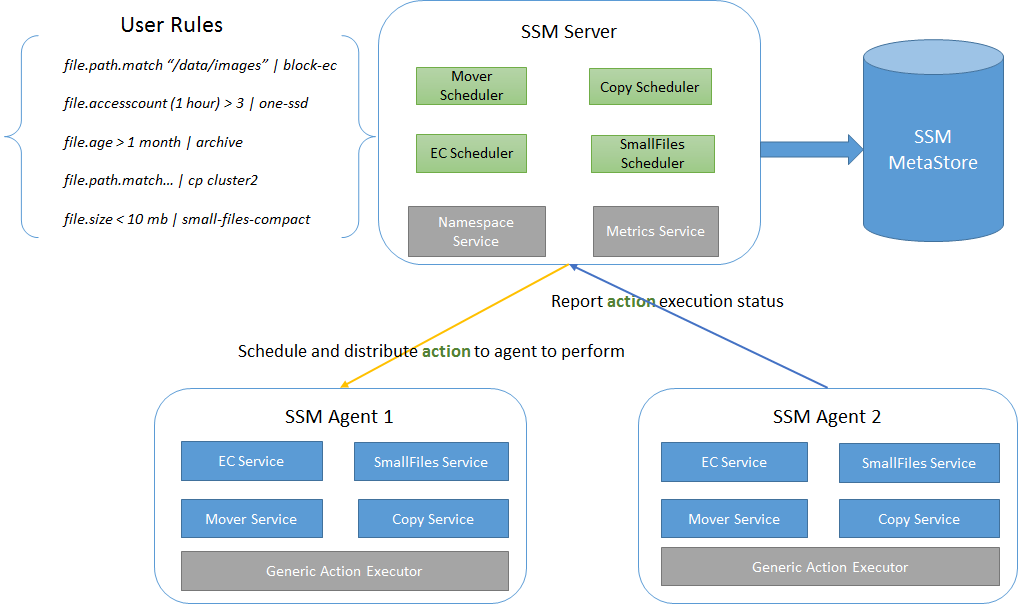
1. 数据格式不仅仅是纯文本，也要考虑更高效的Parquet/ORC等文件格式
2. 在Hadoop中，可以运行多种计算模块包括流计算、内存计算及批计算等多种任务，因此需要考虑不同计算框架对存储的需求

但是Hadoop文件系统作为大数据平台的数据存储基础设置存在很大的问题：其天生为大文件及批处理而设计，对存储海量小文件支持度不高；缺乏数据中心的容灾；不能灵活的将数据在多种存储介质上进行管理，对多种计算框架对存储的需求不能满足等。为了解决这些问题，Intel开源了SSM工具，其自动收集HDFS中数据使用及系统状态数据，基于分析进行数据的复杂管理来优化存储效率，主要功能包括：

1. 根据数据的访问冷热程度自动选择存储介质，实现数据在HDFS的异构存储及缓存的移动
2. 支持Erasure Code，减少使用HDFS 三副本的存储容量
3. 支持小文件合并
4. 支持数据的透明复制及灾备，在不同的Hadoop集群之间进行数据管理
5. 支持数据的透明压缩

# SSM系统架构

SSM系统架构图如下所示：

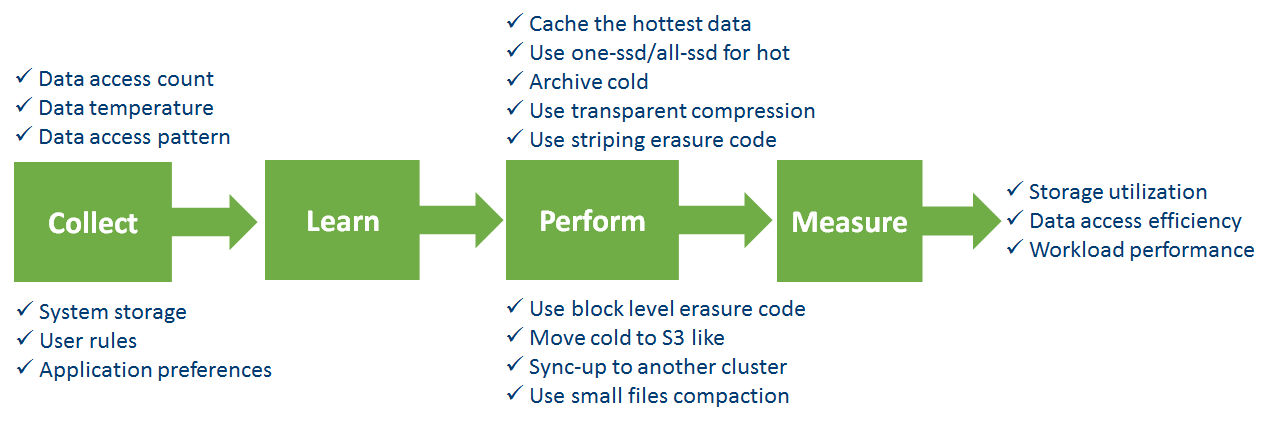


1. SSM MetaStore，使用数据库存储文件管理policy元数据，支持多种数据库包括MySQL，Embed数据库TiDB/TiKV等
2. SSM Server，从Hadoop集群中获取文件存储及访问信息，并根据MetaStore中定义的文件管理Policy，生成执行命令
3. SSM Agent，接收SSM Server分配的命令并执行
4. Web，还提供基于Zeppelin的UI，方便用户定义policy及文件服务的管理

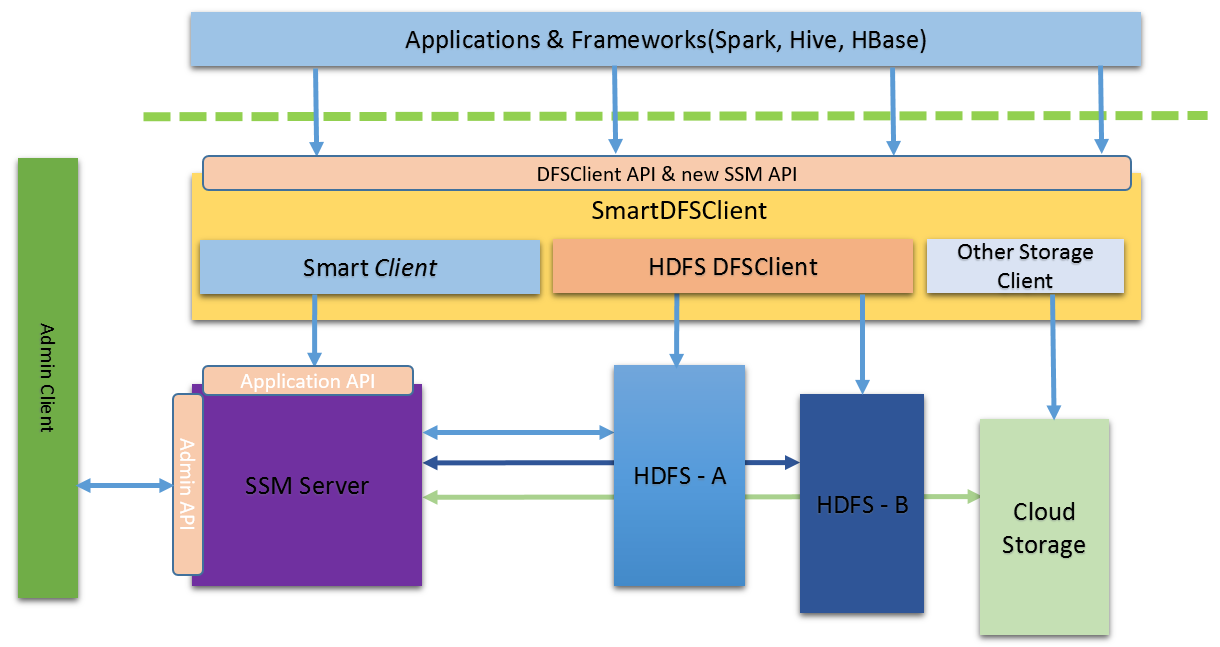
SSM是典型的Master-Slave架构，并且有内置分布式数据库，用来存储文件状态和用户指定的规则。

## SSM基本概念

下图是SSM系统的执行序列图：



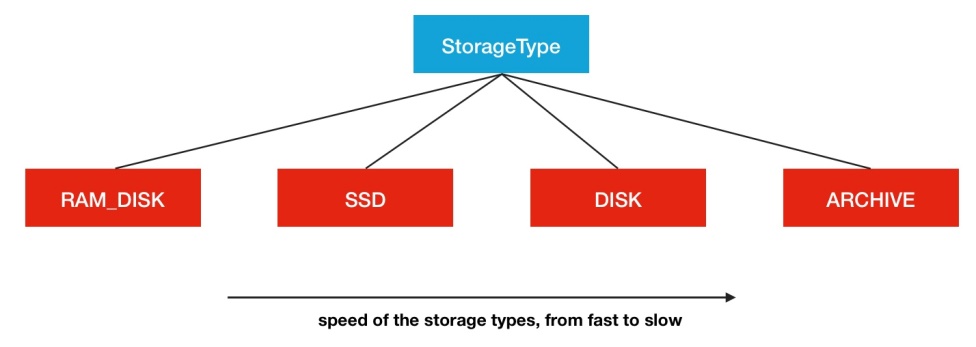
SSM Server通过API获取数据访问数目、数据热度及数据访问模式，获取系统存储。用户通过Web UI配置User rules，Server根据这些信息生成数据管理策略，包括缓存最热数据、归档冷数据、使用透明压缩、Erasure Code等。SSM在大数据生态系统中的位置如下：



SSM是存储和计算的中间层，为上层的计算框架提供全新的客户端API，同时封装好HDFS的原生API，在上层支持多种计算框架的同时，底层支持多种异构的存储系统。SSM提供统一的客户端，访问多Hadoop集群、云存储及SSM Server。

## **使用场景**

SSM的主要使用场景是将数据根据使用冷热程度选择不同的存储介质，异构存储可以根据各个存储介质读写特性发挥各自的优势，例如冷数据，可以采用容量大，读写性能不高的存储介质存储，比如最普通的Disk磁盘。而对热数据而言，可以采用SSD的方式进行存储，这样可以保证高效的读性能，在速率上做到十倍或百倍与普通的磁盘读写速度。HDFS异构存储支持的存储介质类型如下图：



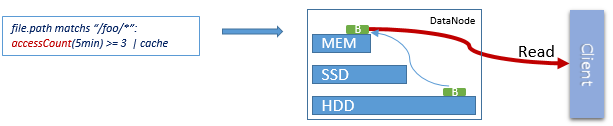
其中ARCHIVE存储并没有特定指哪种存储介质，主要是存储低廉高密度存储（PB级别存储，耗电低），该存储适用于冷数据的存储,比如最普通的Disk磁盘。

HDFS的异构存储是受Storage Policy控制的，在HDFS中定义了5种不同的Storage Policy，列表如下：

| **PolicyID** | **PolicyName** | **Block Placement (n  replicas)** | **Fallback storages for creation** | **Fallback storages for replication** |
| --- | --- | --- | --- | --- |
| 15 | Lazy\_Persist | RAM\_DISK: 1, DISK: *n*-1 | DISK | DISK |
| 12 | All\_SSD | SSD: *n* | DISK | DISK |
| 10 | One\_SSD | SSD: 1, DISK: *n*-1 | SSD, DISK | SSD, DISK |
| 7 | Hot (default) | DISK: *n* | <none> | ARCHIVE |
| 5 | Warm | DISK: 1, ARCHIVE: *n*-1 | ARCHIVE, DISK | ARCHIVE, DISK |
| 2 | Cold | ARCHIVE: *n* | <none> | <none> |

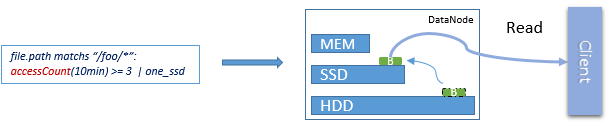
当数据定义为Hot Policy时，block全部复本都在磁盘上，计算和存储都在节点上;Cold，存储到计算能力差的节点上，这些数据以后很少会被使用，数据需要做归档，冷数据所有复本都存储到ARCHIVE中;Warm, 数据的复本部分在ARCHIVE上，部分在DISK上。

1. **缓存热数据**

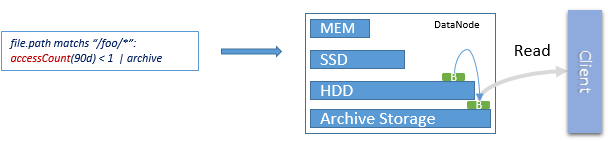


5分钟内访问次数超过3次，则将文件放到缓存中。

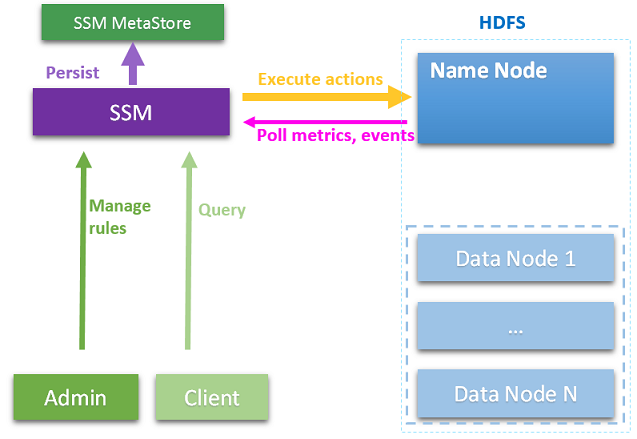
1. **将热数据移动到快速存储**



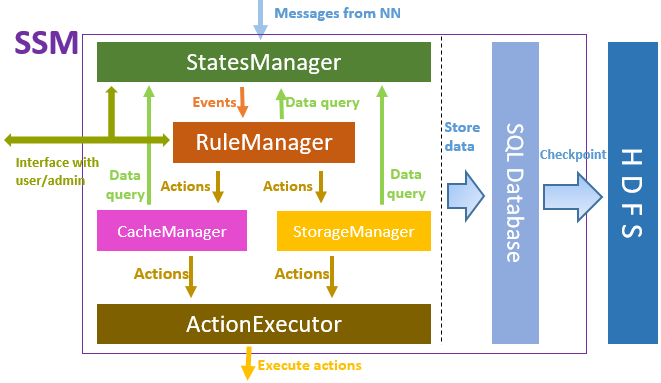
1. **冷数据归档**，在Hadoop中大部分数据在使用后不再被访问，可以将这些数据归档



## **SSM执行流程**

****

SSM从NameNode中拉取Metric信息，SSM分析Metric，当符合定义的rule时生成对应的操作。SSM包括5个主要组件：



1. StatesManager，从NN中收集metric及events，并维护这些数据
2. RuleManager，管理Rule，并根据StatesManager中维护的数据执行这些rules
3. CacheManager，产生缓存相关的操作，并传送给ActionExecutor
4. StorageManager，产生存储相关的操作，并传送给ActionExecutor
5. ActionExecutor，执行具体的操作

以上产生的数据通过SQL存储到数据库中。

# SSM的安装及使用

1）下载源代码并编译

*git clone <https://github.com/Intel-bigdata/SSM.git>*

*mvn package -Pdist,web,native -Dtar -DskipTests*

安装包在smart-dist/target目录下，安装包为smart-data-1.5.0-SNAPSHOT.tar.gz，解压后目录结构如下所示：

.

├── bin //启动脚本

│   ├── common.sh

│   ├── functions.sh

│   ├── smart

│   ├── smart-init.sh

│   ├── start-smart.sh

│   └── stop-smart.sh

├── conf //配置文件

│   ├── agents

│   ├── configuration.xsl

│   ├── druid.xml

│   ├── hazelcast.xml

│   ├── interpreter-list

│   ├── log4j.properties

│   ├── shiro.ini.template

│   ├── smart-default.xml

│   ├── smart-site.xml

│   └── zeppelin-site.xml

|── dist //SSM Web UI端

├── lib //产品依赖库

├── LICENSE.txt

├── logs

├── notebook // zeppelin notebook

│   └── 2CM9DW8NW

└── README.md

1. 配置

* smart-site.xml，配置Hadoop NN Rpc地址及配置文件目录

*<property>*

*<name>smart.dfs.namenode.rpcserver</name>*

*<value>hdfs://cmhhost1.novalocal:8020</value> //rpc地址*

*</property>*

*<property>*

*<name>smart.hadoop.conf.path</name> //配置文件目录*

*<value>file:///etc/hadoop/conf</value>*

*</property>*

配置数据库druid.xml

*<properties>*

*<entry key="url">jdbc:mysql://cmhhost3.novalocal:3306/test</entry>*

*<entry key="username">root</entry>*

*<entry key="password">123456</entry>*

*</properties>*

配置agents

*cmhhost3.novalocal*

其他为默认配置

1. SSM启动，命令执行如下：

*bin/start-ssm.sh*

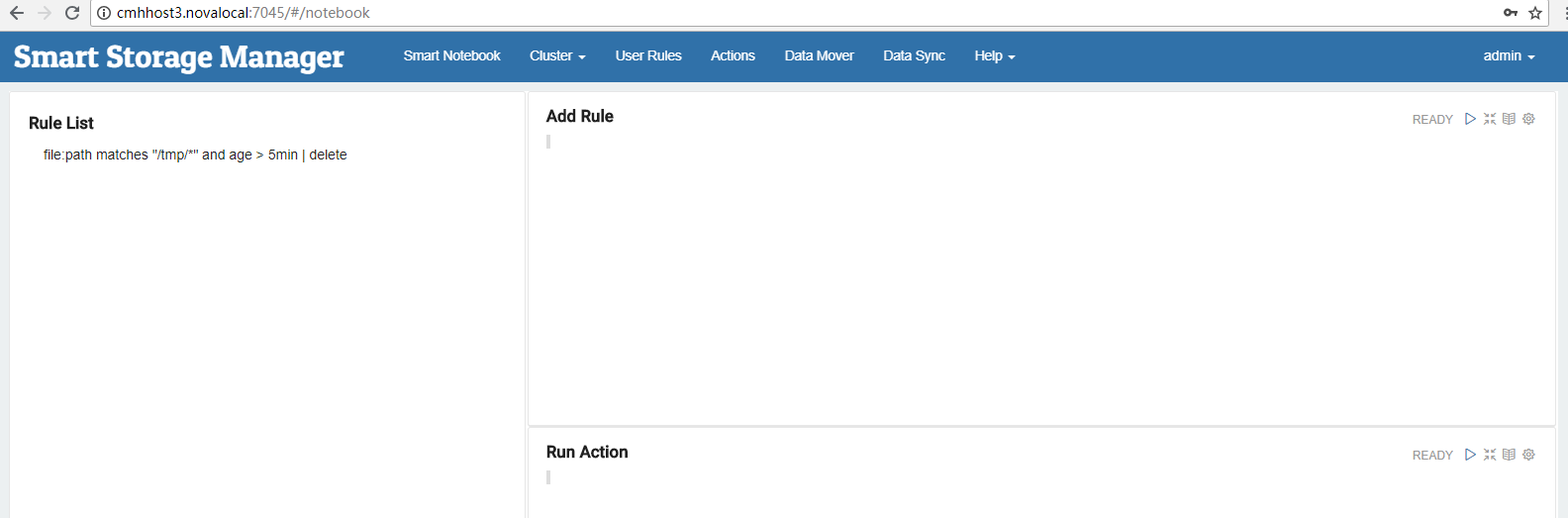
启动的进程如下：

*$ jps*

*29075 SmartAgent*

*28873 SmartDaemon*

Web UI的默认端口为7045，打开页面如下：



执行后，在数据库中写入的数据如下：

*mysql> show tables*

*+-----------------------------------------+*

*| Tables\_in\_test |*

*| access\_count\_table |*

*| action |*

*| backup\_file |*

*| blank\_access\_count\_info |*

*| cached\_file |*

*| cluster\_config |*

*| cluster\_info |*

*| cmdlet |*

*| datanode\_info |*

*| datanode\_storage\_info |*

*| ec\_policy |*

*| file |*

*| file\_diff |*

*| file\_state |*

*| global\_config |*

*| rule |*

*| small\_file |*

*| storage |*

*| storage\_hist |*

*| storage\_policy |*

*| sys\_info |*

*| xattr |*

*+-----------------------------------------+*

1. 添加rule及执行，在Web上添加Rule

*file:path matches "/tmp/\*" and age > 5min | delete*

*//查看目录tmp下的文件，删除5分钟之前的数据*

在Add Rule窗口中添加rule



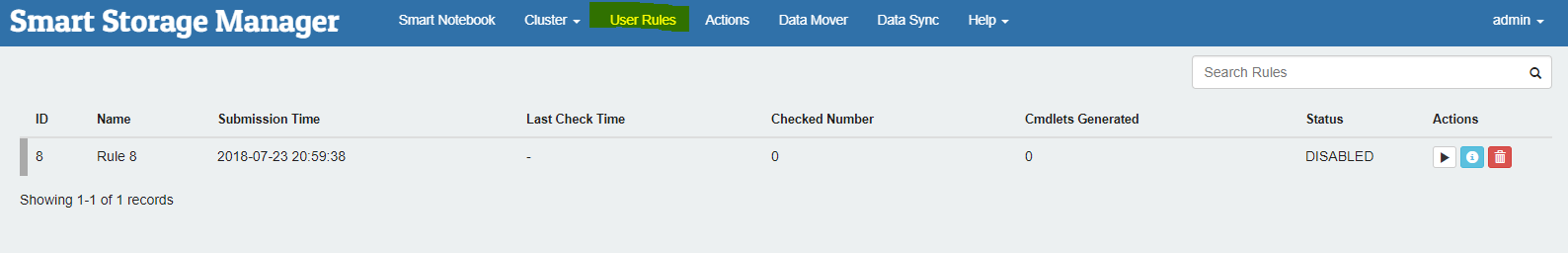
点击READY后添加到User Rules数据库中，查询rule数据库表如下：

*id | name | state | rule\_text | submit\_time| last\_check\_time | checked\_count | generated\_cmdlets |*

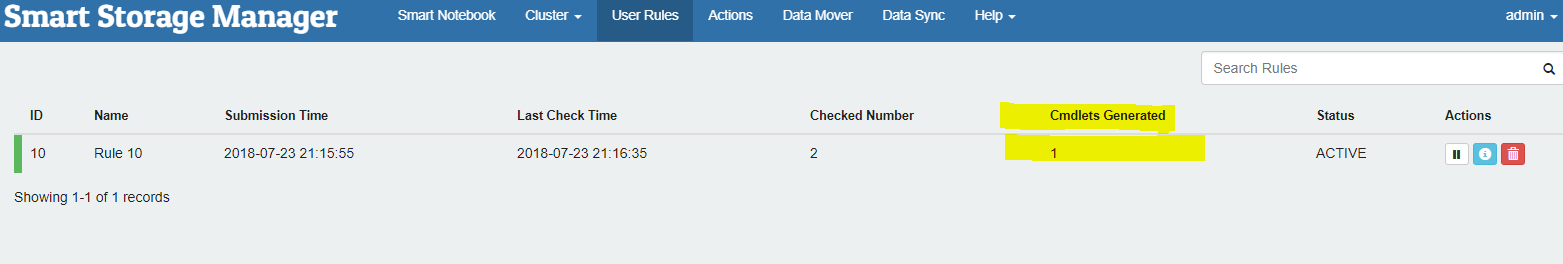
*| 7 | NULL | 4 | file:path matches "/tmp/\*" and age > 5min | delete | 1532335931739 |*

*1532339449956 | 235 | 2 |*

在页面Tab下显示如下：



点击Active，则生效，观察如下：



执行一次，在agent中接收到命令，然后执行，将/tmp/下的文件删除，查看表cmdlet，可以看到生成的命令：

*myql> select \* from cmdlet;*

*+-----+-----+------+-------+------------+---------------+--------------------+*

*| cid | rid | aids | state | parameters | generate\_time | state\_changed\_time |*

*+-----+-----+------+-------+------------+---------------+--------------------+*

*| 0 | 5 | 0 | 10 | delete | 1532334915554 | 1532334917271 |*

*......*

mysql> select \* from action; //执行的操作

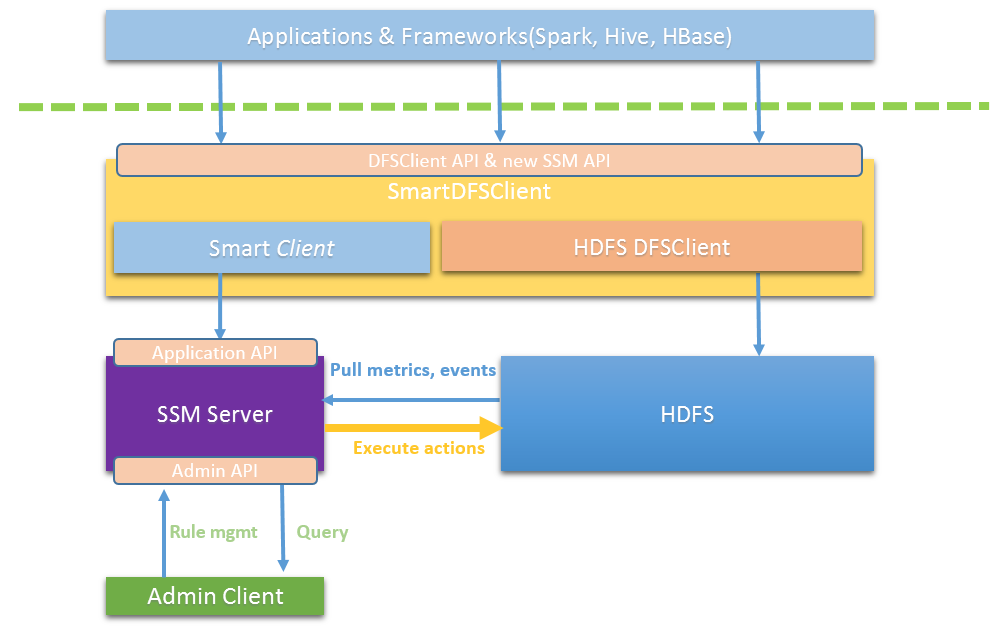


# SSM API && APIS

SSM的API有两种类型：

1. Admin API，Hadoop集群管理用户使用Admin API管理文件的rules，包括文件rule的创建/删除/list/Update等
2. Application API，运行在HDFS上的操作，包括文件的移动/archive/cache等操作

Admin及Application API可能对文件同时进行操作，Application API的优先级比Admin API高。这些接口通过RPC或者RESTful接口提供：



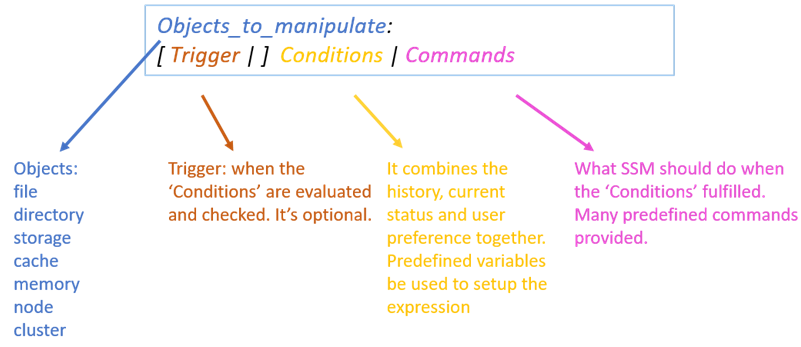
Admin API通过定义文件操作Rule来管理HDFS，在上例中文件的删除Rule定义如下：

*file: path matches "/tmp/\*" and age > 5min | delete*

*匹配/tmp目录下所有文件，当文件超过5分钟，则删除*

## **Rule的定义格式**

其定义格式如下：



Rule的定义包含4个部分：

1. Object to manipulate，包括：file/directory/storage/cache/memory/node/cluster，以文件Object为例

|  |  |  |
| --- | --- | --- |
| Object | Description | Example |
| file | Files | File with path mathes “/fooA/\*dat” |

1. Trigger，文件操作的触发时机，该部分是可选

|  |  |  |
| --- | --- | --- |
| Format | Description | Example |
| at <time> | 在给定时间执行Rule | -at  “2017-07-29 23:00:00”  - at now |
| every <time interval> | 给定频率执行Rule | -every 1min |
| from <time> [To <time>] | 在时间范围内执行rule | - every 1day from now  - every 1min from now to now + 7day |

1. Conditions

|  |  |  |
| --- | --- | --- |
| Ingredient | Description | Example |
| Object Property | Object参数的条件 | -length > 5MB |
| Time | - “yyyy-MM-dd HH:mm:ss:ms”  - Predefined  - Time + Time Interval | - “2017-07-29 23:00:00”  - now  - now + 7day |
| Time Interval | - Digital + unit  - Time – Time  - Time Interval + Time Interval | - 5ms, 5sec, 5min, 5hour, 5day  - now - “2016-03-19 23:00:00”  - 5hour + 5min |
| File Size | -Digital + unit | - 5B, 5kb, 5MB, 5GB, 5TB, 5PB |
| String | Start and ends with “, support escapes | - “abc”, “123”, “Hello world\n” |
| Logical Operator | and,or,not |  |
| Digital Operator | +, -, \*, /, % |  |
| Compare | >, >=, <, <=, ==, != |  |

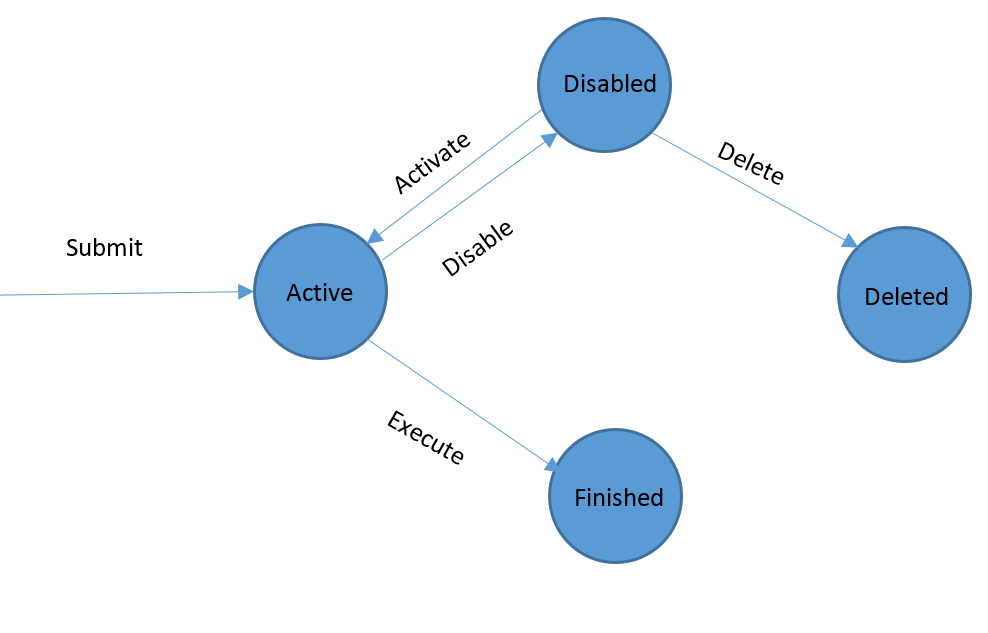
1. Commands

|  |  |
| --- | --- |
| 命令 | 描述 |
| allssd | 将文件移动到SSD存储 |
| alldisk | 将文件移动到磁盘存储 |
| append | 将length或者bufsize的文件append到文件中 |
| archive | 文件归档 |
| cache | 将文件缓存到HDFS中 |
| checkstorage | 展示文件的存储类型 |
| compact | 将两个合并到containerFile中 |
| concat | 将src文件cat到dst文件中 |
| copy | 将src文件复制到dst |
| copy2s3 | 将src文件复制到A3存储中 |
| delete | 删除文件 |
| echo | 输出$message |
| list | 列举src文件中所有文件 |
| merge | 将两个文件进行merge |
| onedisk | 将文件的一个副本移动到disk |
| onessd | 将文件的一个副本移动到SSD |
| ramdisk | 将文件移动到RAM\_DISK |
| read | 读取$size的文件内容 |
| rename | 文件重命名 |
| truncat | 文件的truncat操作，长度为$length |
| truncat0 | 文件truncat为0 |
| uncache | 从HDFS Cache中uncache文件 |
| uncompact | 将containerFile还原到原文件 |
| write | 向文件中随机写入到数据 |
| sleep | 暂停给定时间 |
| sync | 将文件同步到远程cluster中 |
| user defined actions | 用户自定义action |

1. Object Property，没有对象都有参数，以file object为例

|  |  |  |
| --- | --- | --- |
| Object | Property | Description |
| file | age | 文件存在时间，例如5min |
| blocksize | 文件的block大小 |
| inCache | 文件是否在cache中 |
| isDir | 文件是否为目录 |
| length | 文件的长度 |
| path | 文件路径 |
| mtime | 文件的修改时间 |
| unsynced | 文件是否同步 |
| storagePolicy | 文件的存储策略 |
| accessCount(Time Interval) | 在时间间隔内的访问次数 |
| accessCountTop(interval,N) | 在时间间隔内top N的访问次数 |
| accessCountBottom(interval,N) | 在时间间隔内under N的访问次数 |
| accessCountTopOnStoragePolicy(interval,N,$StoragePolicy") |  |
| accessCountBottomOnStoragePolicy(interval,N,$StoragePolicy") |  |

在系统中Rule有4中状态：active,disabled,finished及deleted，转换图如下所示：



## **Rule管理API**

Rule的管理接口有两种：RPC和RESTful HTTP接口，RPC的接口如下：

* long submitRule(String rule)， 提交到rule
* long submitRule(String rule,RuleState initState)，提交rule并指定初始状态，可以为active或者disabled
* void checkRule(String rule)，验证rule
* RuleInfo getRule(long ruleId)，根据ruleId或者Rule信息
* List<RuleInfo> listRules，列举系统中的rule
* void deleteRule(long ruleId,boolean dropPendingCommands)，删除rule
* void enableRule(long ruleId)，启动rule
* void disableRule(long ruleId,boolean dropPendingCommands)，禁用rule

参考链接：

https://github.com/Intel-bigdata/SSM/blob/trunk/docs/admin-user-guide.md

https://github.com/Intel-bigdata/SSM

http://www.sohu.com/a/165817614\_779707