HDFS Smart Storage Management

Towards Higher Storage Efficiency

Wei Zhou

Apache Big Data Europe 2016

Outline

- Motivation
- Architecture
- Design
- Rule
- Case Study
- Summary

Motivation

- Data to be processed and stored boosts
  - ✓ Internet of Things
  - ✓ Real time stream processing
  - ✓ Online Analytical Processing
  - ✓ Artificial Intelligence / Deep Learning

- Data needs to be processed in time
  - ✓ From data been generated to been processed
  - ✓ Stored with complex format

Motivation

Support for more scenarios

- File size
- Temperature: hot and cold
- Workloads: on-line query / off-line analysis

Motivation

Object storage HDFS-7240

- Targets at:
  - ✓ Billions of objects
  - ✓ Vary for from KB level to tens of MB
  - ✓ Reliability, consistency and availability
- Object store. No file metadata, K/V based API
- Supported in Amazon S3, Azure, Aliyun, …

Motivation

Hardware

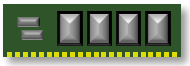
Network bandwidth increases

- 10Gbps network is the mainstream
- 40Gbps or even 100Gbps is on the way

Motivation

Hardware

- More memory
- Storage device
  - ✓ Cheaper. History data
  - ✓ Faster. NVMe and 3D XPoint® Technology
- Different types of storage used in HDFS



Motivation

Software

Facility	Target	Using
Cache	Performance	Call API explicitly
Heterogeneous Storage Management	Performance Cost saving	Call API explicitly
Erasure Coding	Space saving	Call API explicitly
Mover	Maintain	Call CLI explicitly
Storage Policy Satisfier	Maintain	Call API explicitly
DiskBalancer	Maintain	Call API explicitly

But this is not the end of the story!

Motivation

But it remains a BIG challenge to identify…

Motivation

Something that can handle these issues automatically and smartly by using the right facilities at the right time.

Motivation

Key to these questions

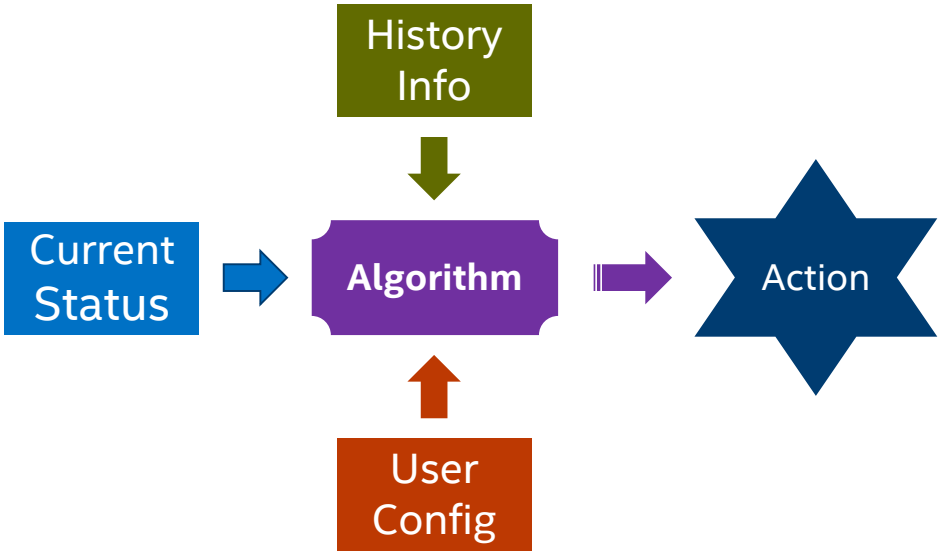
- ✓ sense the data temperature timely
- ✓ predicate the temperature change
- ✓ deal with the change
- ✓ evaluate a storage device's efficiency

Motivation

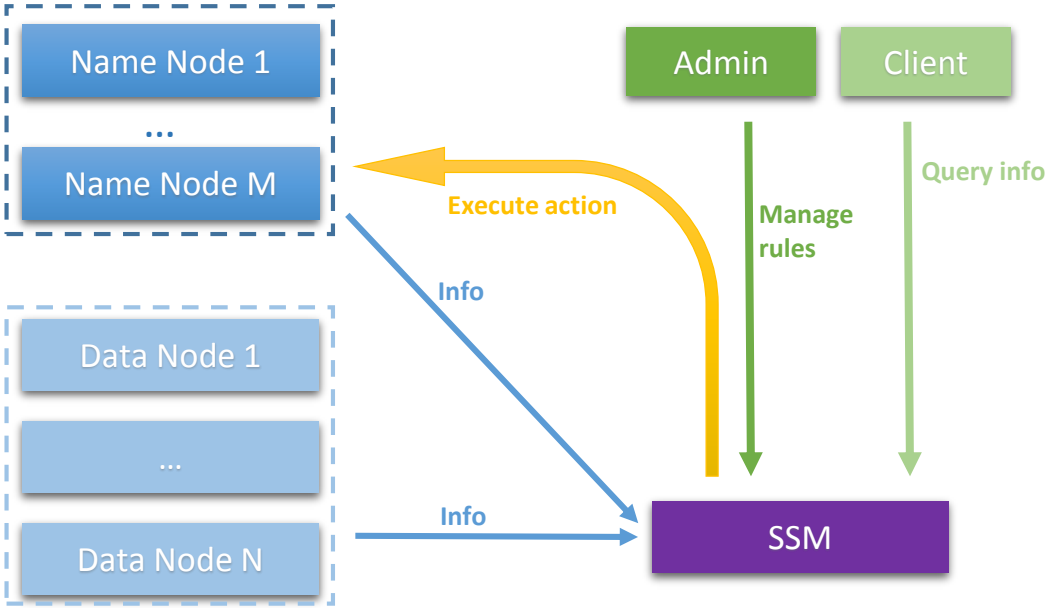
To solve these question, we have to:

- **Learn from history**
  - **access pattern**
- **Aware of current status**
  - **States of resources**
  - **States of data**
- **Respect to users**
  - **Definition and threshold**
  - **Preference**

Motivation



Architecture

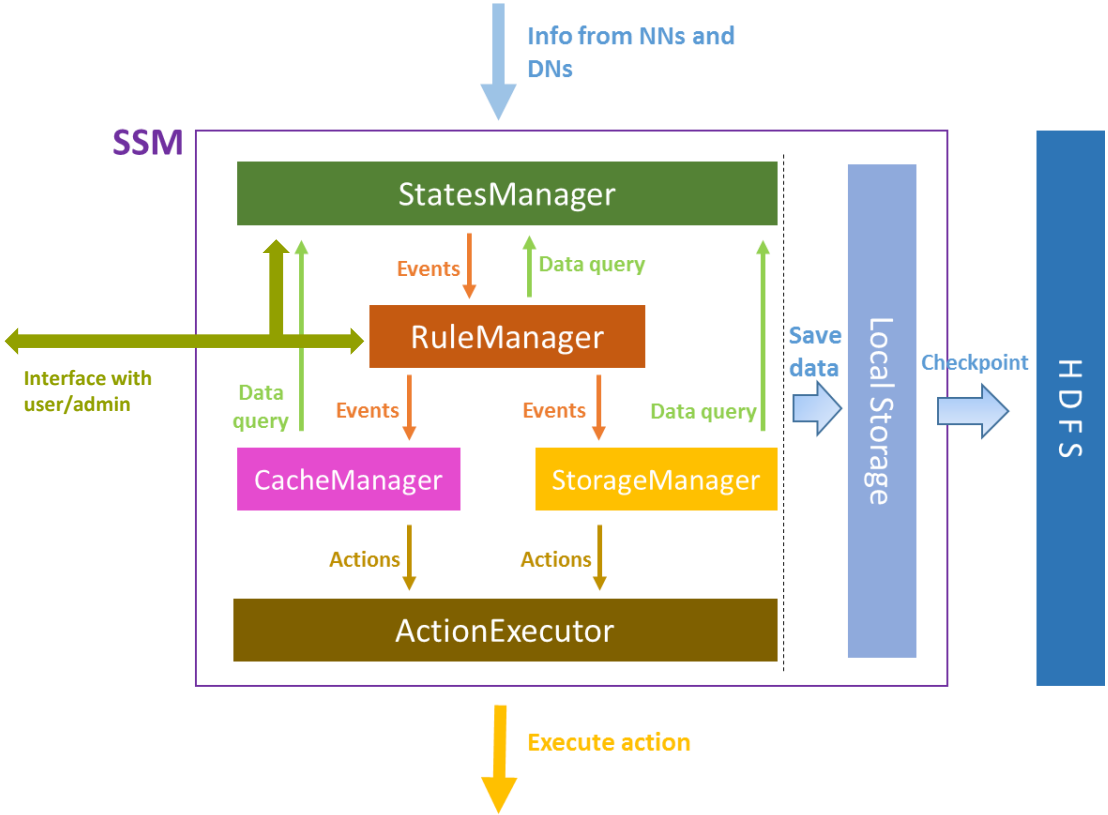


Principle

Before we dive into the detailed design:

- Optional service for HDFS
  - ✓ Run facilities manually may not be allowed
- Should not:
  - ✓ Break the function of cluster
  - ✓ Bring in security issue to the cluster
- Trying to:
  - ✓ Minimize the overhead to the cluster
  - ✓ be simple for porting

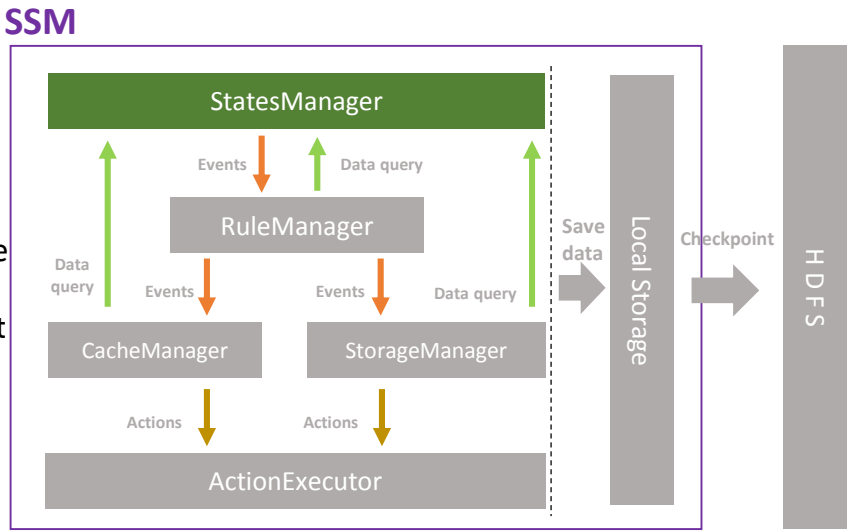
Design


Info from NNs and DNs
SSM
StatesManager
Events
Data query
RuleManager
Interface with user/admin
Data query
Events
Events
Data query
CacheManager
StorageManager
Actions
Actions
ActionExecutor
Save data
Local Storage
Checkpoint
HDFS
Execute action

Design

StatesManager

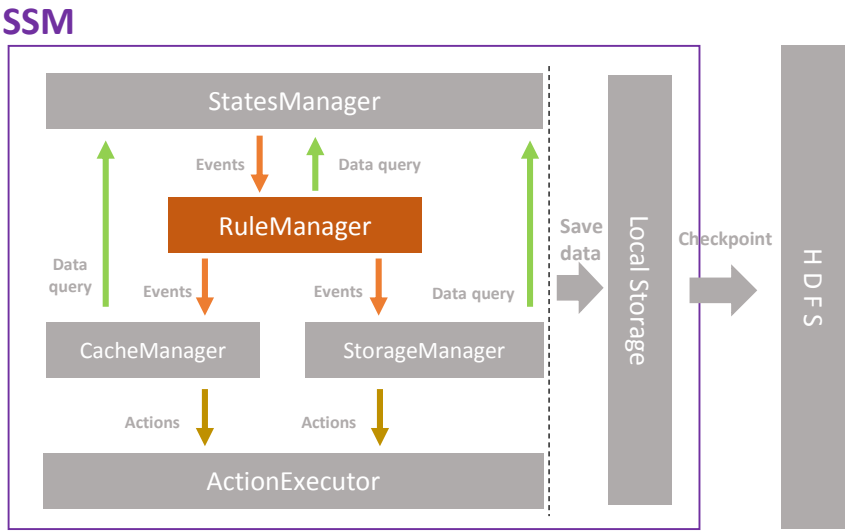
- Historical information. For example, file access history, cache hit statistics, disk throughputs of DataNodes.
- Current status information. E.g. file storage policy, a file is in cache or not. This kind of information is not required to be stored as it can be queried from NameNodes when needed.
- Forward and generate events to RuleManger



Design

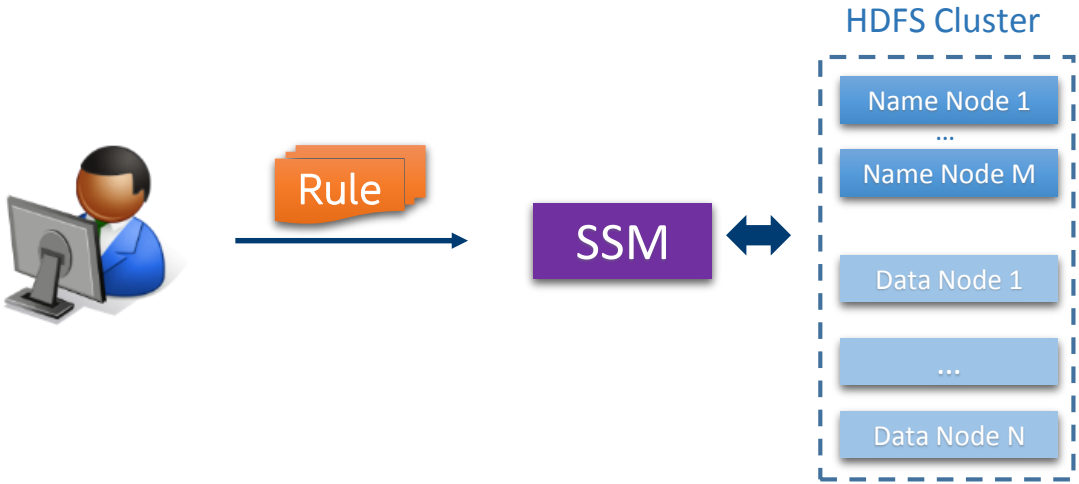
RuleManager

- Parse rules and execute rules
- Explore rule for files without specifying a rule.
  - ✓ Templates



Design

How to use of SSM?



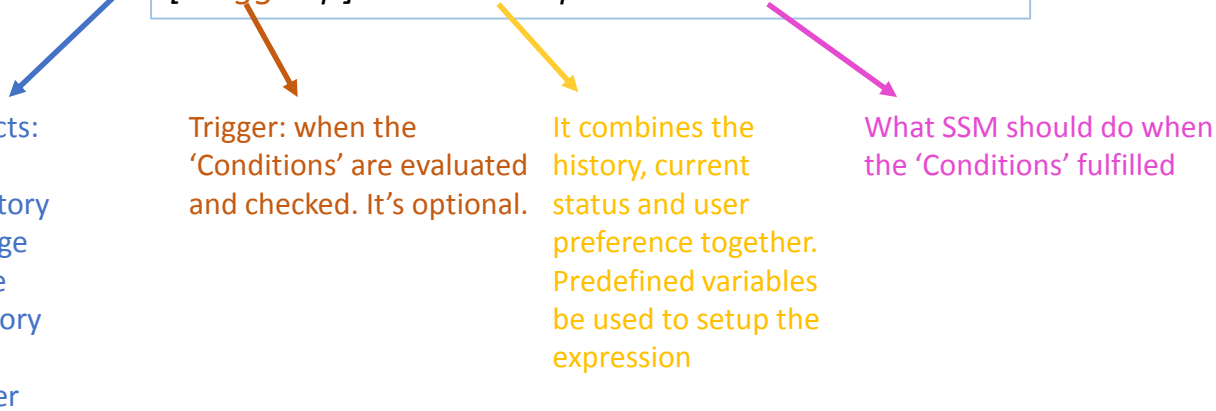
Achieve better performance without modifying upper App logic

Rule

It links history info, current status, user configuration and action together. It's a guide line for SSM to function.

Syntax

```
# comment starts with a '#'
Objects_to_manipulate:
[ Trigger | ] Conditions | Commands
```



Rule

Examples

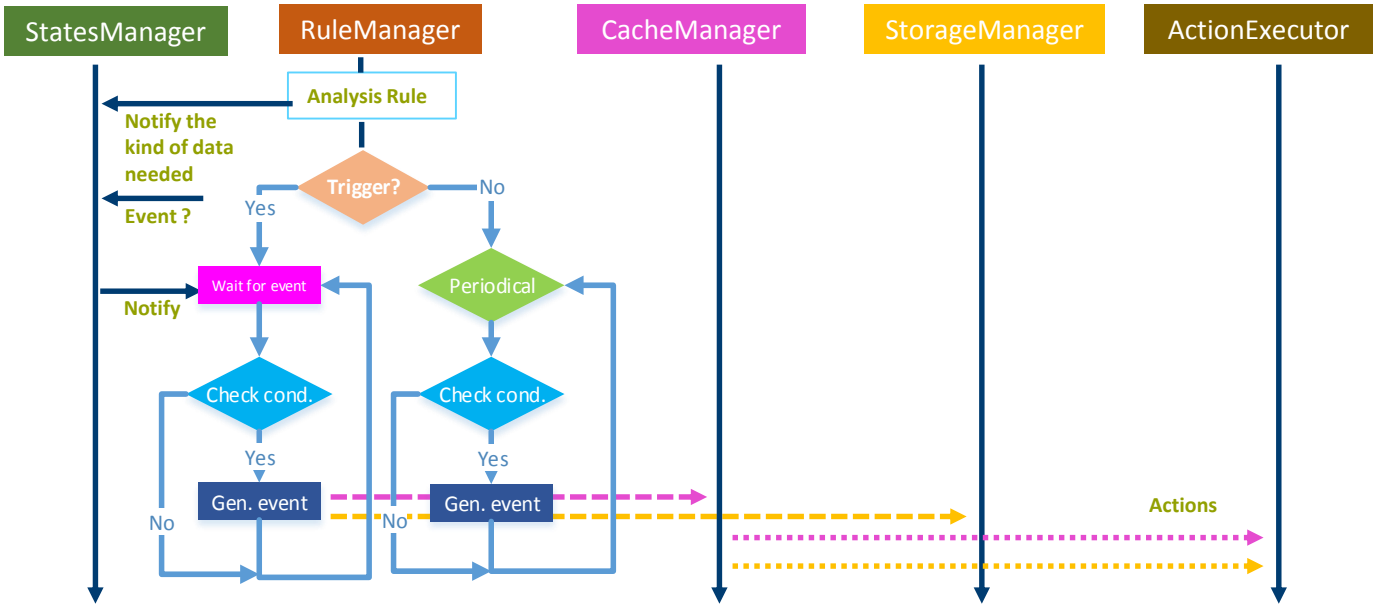
file.path matches "/fooA/abc":*
accessCount(10min) >= 10 | cache

file.path matches "/fooB/":*
age >= 30d | archive

datanode:
every 1:00 | datanode.storageUnbalanceRatio('SSD') > 30 | diskbalance

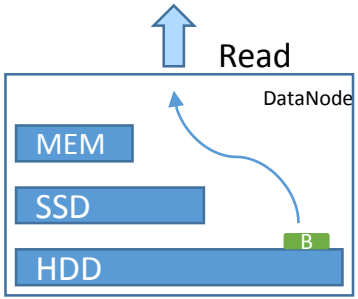
Rule

Execution flow

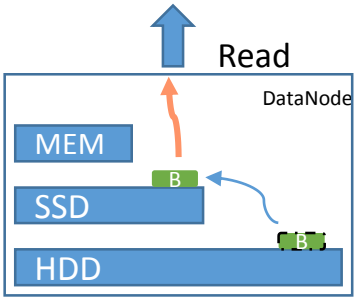


Case Study

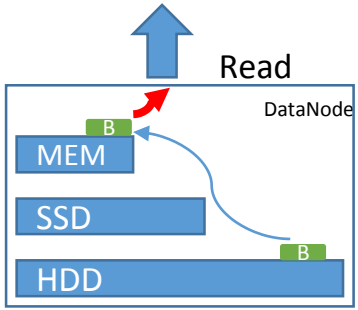
Optimize when getting HOT



file.path matches "/foo/":*
accessCount(10min) >= 3 | mover ONE_SSD



file.path matches "/foo/":*
accessCount(10min) >= 3 | cache



Case Study

Archive COLD data

COLD data: files under directory /foo and age larger than 30 days

Without SSM
It's hard to implement! ☹☹

With SSM

file.path matchs "/foo/":*
age > 30d | archive

Archive when the cluster is in low load

Case Study

Archive COLD data

COLD data: files under directory /foo and not been read for more than 3 times in last 30 days

Without SSM
It's hard to implement! ☹☹

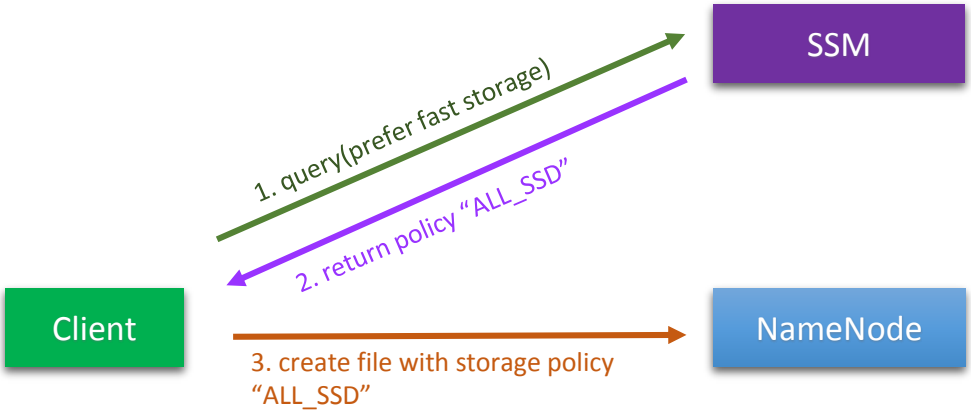
With SSM

file.path matchs "/foo/":*
accessCount(30d) < 3 | archive

Archive when the cluster is in low load

Case Study

Optimization on write with fast storage



Status

- The discussion is continuing on

- Prototype undergoing

  Implementation for the 3 use cases

  - ✓ Archive cold data
  - ✓ Move hot data to fast storage
  - ✓ Cache hot data

Status

- Enhance HDFS cache for partial caching
- Block-level statistics and optimization
- Extend EC for data archive usage

Summary

We introduce in an mechanism to optimize the efficiency of HDFS cluster:

- Rule-based engine
- State-aware management
- Automation
- Provide an unified interface to user
- Flexible
- Tune HDFS to fit application behaviors

JIRA: HDFS-7343

Any suggestions or participations will be appreciated!


intel

Legal Disclaimer