Improve YARN shared cache support for LinuxContainerExecutor

Chris Trezzo
YARN-5727

Problem

When running LinuxContainerExecutor in a secure mode (yarn.nodemanager.linux-container-executor.nonsecure-mode.limit-users set to false), all localized files are owned by the user that owns the container which localized the resource. This presents a problem for the shared cache when a YARN application requests a resource to be uploaded to the shared cache that has a non-public visibility. The shared cache uploader (running as the node manager user) does not have access to the localized files and can not compute the checksum of the file or upload it to the cache. In this document we will discuss various solutions to this problem, all of which should ideally satisfy the following three requirements:

1. Localized files should still be safe/secure. Other users that run containers should not be able to modify, or delete the publicly localized files of others.
2. The node manager user should be able to access these files for the purpose of checksumming and uploading to the shared cache without being a privileged user.
3. The solution should avoid making unnecessary copies of the localized files.

Solution

Add a separate scstaging directory in the /yarn/local directory structure of the node manager that will contain localized resources for each application that have public visibility and are set to be uploaded to the shared cache. This directory will have similar permissions to the other cache directories (i.e. globally readable and executable):

```
/yarn/local/                              drwxr-xr-x.:nmuser:group1
     /filecache                           drwxr-xr-x.:nmuser:group1
     /usercache                           drwxr-xr-x.:nmuser:group1
     /scstaging                           drwxr-xr-x.:nmuser:group1
```

Within the scstaging directory, there will be a file cache that is structured in a similar way to the application cache within the usercache:

```
.../scstaging/<appId>/                    drwxr-xr-x.:nmuser:group1
     /filecache                           drwxr-xr-x.:nmuser:group1
          /11/                            drwxr-xr-x.:nmuser:group1
               /11/resource1.txt          -r-xr-xr-x.:nmuser:group1
          /12/                            drwxr-xr-x.:nmuser:group1
               /12/job.jar                -r-xr-xr-x.:nmuser:group1
```

The biggest difference is that the visibility of resources in this cache will be public and the application directories will be owned, created, and deleted by the node manager user as part of normal application setup/cleanup. This creates a public read-only directory for localized resources that has the same life cycle of the corresponding application. You can think of it as a cross between the public cache and the application cache.

Before this change, all resources with a public visibility would be localized by the public localizer and stored in the public cache. After this change, resources that are set to be uploaded to the shared cache and have a public visibility are localized by a new scstaging localizer and stored in the application specific scstaging directory instead of the public cache. This allows the shared cache uploader service to checksum and uploade resources without these resources lingering on the machine until the public cache hits a target size, or cause additional churn in the public cache if it is already full.

Potential affected parts of the codebase

1. Node manager state store
2. Upgrades/downgrades
3. Localization

    a. ContainerLocalizer
    b. ResourceLocalizationService
        i. New LocalResourceTracker for sharedcache
    c. LinuxContainerExecutor
4. SharedCacheUploadService
5. SharedCacheUploader

Other considered solutions

Public cache approach

One way to avoid this issue is to ensure that any local resource that should be uploaded to the shared cache has a public visibility. Publicly visible resources localized by YARN are put in the public local cache. The files in the local public cache are world readable and owned by the node manager user. With the resources in the public cache, the shared cache uploader can easily access them for uploading and checksumming.

This will require YARN applications to ensure that local resources have a public visibility set and the resource is publicly visible. For example, in MapReduce, this would require making resources that need to be uploaded to the shared cache, publicly visible even though they are in the staging directory. Additionally, changes at the YARN level could be made to fail an attempt to localize a resource that should be uploaded to the shared cache that does not have a public visibility. This would stop the silent failures when trying to upload the resource to the shared cache when using the LinuxContainerExecutor.

On the positive side, this approach is the least invasive and it solves the problem without relying on platform specific features. That being said, there are some disadvantages to this approach. Currently, the majority of resources localized by MapReduce wind up in the Application level cache. This means the life cycle of these local resources is tied to the life cycle of the application. If these resources are moved to the public cache, they will stay in the cache until the cache needs to delete them to maintain its target size. Additionally, these resources could create additional churn that would bounce more heavily used resources from the public cache, adversely affecting the public cache hit rate.

ACL-based approach

If we do not want to use the public local cache for all of the resources, we could keep the basic file permissions for the resources the same and instead leverage ACLs. In this solution, each resource that should be uploaded to the shared cache would have an acl added to it via the ContainerLocalizer that would give read permissions to the node manager user. This would allow the shared cache uploader to checksum the local resource and upload it to the public

shared cache. The downside of this approach is that it relies on the presence of ACLs and added configuration.

Individual user approach (non-solution)

Another approach would be to leave the local resources with whatever visibility they have, but do the checksuming and uploading of resources to the shared cache as the user that owns the resource. This would require consolidating the shared cache uploader logic into the ContainerLocalizer and potentially adding changes to the native container-executor code. This approach has a few disadvantages. First, the contents of the shared cache could then be modified by users that own the various files, this violates the requirement that contents in the shared cache should be protected from modification. Second, this would require the SCM and the cleaner to run as an HDFS privileged user so that both of these services could properly clean the cache.

NodeManager Directory Structure for Localization

There are three major directories in the node manager directory structure: filecache, usercache, nmPrivate. The first two (filecache and usercache) are part of the YARN local cache for resource localization. The last directory (nmPrivate) is used for caching application specific files, such as the container launch scripts, and will not be addressed in this document.

```
/yarn/local/          drwxr-xr-x.:nmuser:group1
     /filecache       drwxr-xr-x.:nmuser:group1
     /usercache       drwxr-xr-x.:nmuser:group1
```

The filecache (and publicly-scoped resources)

The filecache directory is used to cache resources that have a publicly-scoped visibility. Each resource is cached in it's own subdirectory. These subdirectories are named using a semi-unique resource id generated by incrementing an atomic long. These files and directories are managed by the public localizer and owned by the node manager user.

```
/yarn/local/filecache/          drwxr-xr-x.:nmuser:group1
     /11/                       drwxr-xr-x.:nmuser:group1
     /11/resource1.txt          -r-xr-xr-x.:nmuser:group1
     /12/                       drwxr-xr-x.:nmuser:group1
     /12/job.jar                -r-xr-xr-x.:nmuser:group1
```

The usercache

The usercache directory is for resources that are cached with a private or application visibility. Within the usercache directory there is a subdirectory for each user that has localized resources on the node manager. In the case of the LinuxContainerExecutor, these subdirectories are owned by each of the respective users. For example:

```
/yarn/local/usercache       drwxr-xr-x.:nmuser:group1
     /user1                 drwxr-s---.:user1:group2
     /user2                 drwxr-s---.:user2:group2
     /user3                 drwxr-s---.:user3:group2
```

Within each of the user directories are two subdirectories: appcache and filecache. The appcache directory is responsible for caching resources with an application visibility and the filecache directory is responsible for caching resources with a private visibility.

```
/yarn/local/usercache/user1                 drwxr-s---.:user1:group2
     /appcache                            drwxr-s---.:user1:group2
     /filecache                           drwx--x---.:user1:group2
```

Application-scoped resources

Within the appcache directory, resources are stored under a two-level subdirectory system made up of the appId and the filecache. There are other subdirectories in the appId directory, but we will not address those in this document. For example:

```
.../usercache/user1/appcache           drwxr-s---.:user1:group2
     /<appId>/                         drwx------.:user1:group2
     /<appId>/filecache/               drwx------.:user1:group2
```

Within the filecache, resources are stored in separate directories using resource id's in a similar way to the publicly scoped resources. In this case though, the resource id subdirectories and resource files are owned by the owner of the application. For example:

```
.../<appId>/filecache/                 drwx--x---.:user1:group2
     /11/                              drwx------.:user1:group2
     /11/resource1.txt                 -r-x------.:user1:group2
     /12/                              drwx------.:user1:group2
     /12/job.jar                       -r-x------.:user1:group2
```

Privately-scoped resources

Within the usercache/filecache directory, the node manager stores privately-scoped resources in a similar way to public resources. The resources are stored in resource id subdirectories where the directories and files are owned by the user that owns the particular user cache. For example:

```
.../usercache/user1/filecache/        drwx--x---.:user1:group2
      /11/                            drwx------.:user1:group2
      /11/resource1.txt               -r-x------.:user1:group2
      /12/                            drwx------.:user1:group2
      /12/job.jar                     -r-x------.:user1:group2
```

Resources

1. YARN localization deep dive: Part 1, Part 2