June 13, 2017

Policy Based Priority Object Consistency Check and Correction in a Filesystem

Hiro Ramehlal Lalwani
Hewlett Packard Enterprise

Jayasankar Nallasamy
Hewlett Packard Enterprise

Anand A. Ganjihal
Hewlett Packard Enterprise

Policy Based Priority Object Consistency Check and Correction in a Filesystem

Abstract - File storage in cloud deployment and Enterprise needs of File based storage are growing rapidly in today's IT environment. Filesystem enables application programs to store and retrieve user data files, and applications have data dependence on the filesystem to guarantee user with correct and accurate information. Therefore, filesystem has to maintain metadata consistency across many data structures that actually points to filesystem objects such as metadata files, application files and directories. In both offline and online filesystem check, traditional approach has been to check metadata consistency across all the filesystem objects. But we are proposing an alternative method to guarantee all-time consistency on selective filesystem objects.

This disclosure is related to on-demand selective objects consistency check and correction in a filesystem.

One of the fundamental software layer assumption is that the underlying hardware layer forever works as designed, but in reality not valid all the time. Interestingly, due to cost advantage there has been steady increase in demand to use commodity hardware and build critical applications. These commodity systems are unreliable in nature compared to custom built hardware leading to software faults. As a result, Filesystem has to deal with diverse challenges on computation misplays such as bit errors in memory/buses and disks, despite protection methods like ECC, CRC etc. In addition, errors could happen at disk subsystem layers like uninitialized blocks, disk sector failures while RAID being rebuilt, malfunctioning of disk controllers etc. These faults are difficult to detect and get unnoticed by filesystem software and thus are not reported to the system or application (which leads to different metadata in-memory compared to on-disk). In essence, filesystem has to deal with hardware faults. Such non-filesystem errors has the possibility to perform erroneous updates to filesystem metadata structure, potentially resulting in silent metadata corruption. Files system metadata flaws should be discovered as it happens, if not data recovery gets worsen and improbable. Ultimately, it would result in seriously damaged application file causing user data unusable. For Instance, storage applications may maintain their metadata on selective directory and files which need to be consistent all the time. In spite of fact that many vendors have adapted versatile data integrity solutions across various layers of stack but are inadequate to support user preferred granular validation upon need and essential basis. Hence, filesystem has to have a pro-active mechanism to selectively assure on-demand and online file/directory tree level/File system objects consistency thus ascertain application data.

In our solution, we propose policy based object consistency in the File system, where user can choose to set the consistency policy per object, per directory tree, per filesystem metadata files or on the entire file system by setting the policy to root directory. When consistency policy is

set per object, object is validated and corrected for any metadata inconsistencies each time the object is accessed either by external file system syscalls() or by File system internal object access methods.

This level of setting consistency policy gives user the flexibility to choose between set of files or directory tree or certain File system metadata files whose consistency is at most priority before or while accessing the object. For example as shown in Figure 1 below, user can enable consistency policy per file "Foo.txt" or per directory tree "dir-x" or on the entire file system "/root". When "dir-x"'s consistency policy is enabled, then files and subdirectories underneath "dir-x" will inherit the consistency policy such as Files "Boston" and "Andover" under dir-x will automatically get consistency policy enabled.
Methodology to implement concept of consistency policy setting on object bases is as follows:

Procedure *Set_Object_Consistency_Policy* (**input:** File path or File inode number; **input:** Consistency Policy [Metadata | Data])
Begin
Read the on-disk inode and read the on-disk "consistency_policy"

if (on disk consistency policy is not equal to user specific policy) **then // Figure 1 (stage 1, 2 and 3)**
Begin
Acquire object write lock
Update on-disk object consistency policy with user specified "Consistency policy". Update in-memory object consistency policy with user specified "Consistency policy".
Release object write lock
else if (policy is already set) then
Begin
Nothing to be done, just return OK.
End
End

Methodology to implement concept of on-demand object consistency validation as part of object access via File system syscalls based on policy setting on object bases is as follows:

Procedure *Check_object_Consistency_based_on_Policy* (**input:** File in-core inode)
Begin
Consistency Policy = read in-core object consistency policy
If (Consistency Policy set) **then // Figure 1 (stage 4 to 8)**
Begin
Object metadata validation and correction by metadata validation and correction module // stage 6 **If** (data consistency is enabled) **then**

```
            Begin
              Object data checksum validation by Data checksum validation module
              // stage 7
            End
            Execute the user requested File system operation and return.
        else if (Consistency Policy not set) then // Figure 1 (stage 4 and then to stage
        8)
        Begin
            Execute the user requested File system operation and return.
        End
End
```

Methodology to implement concept of on-demand object consistency validation invocation as part of object access via File system syscalls or by any other means which results in object access as follows:

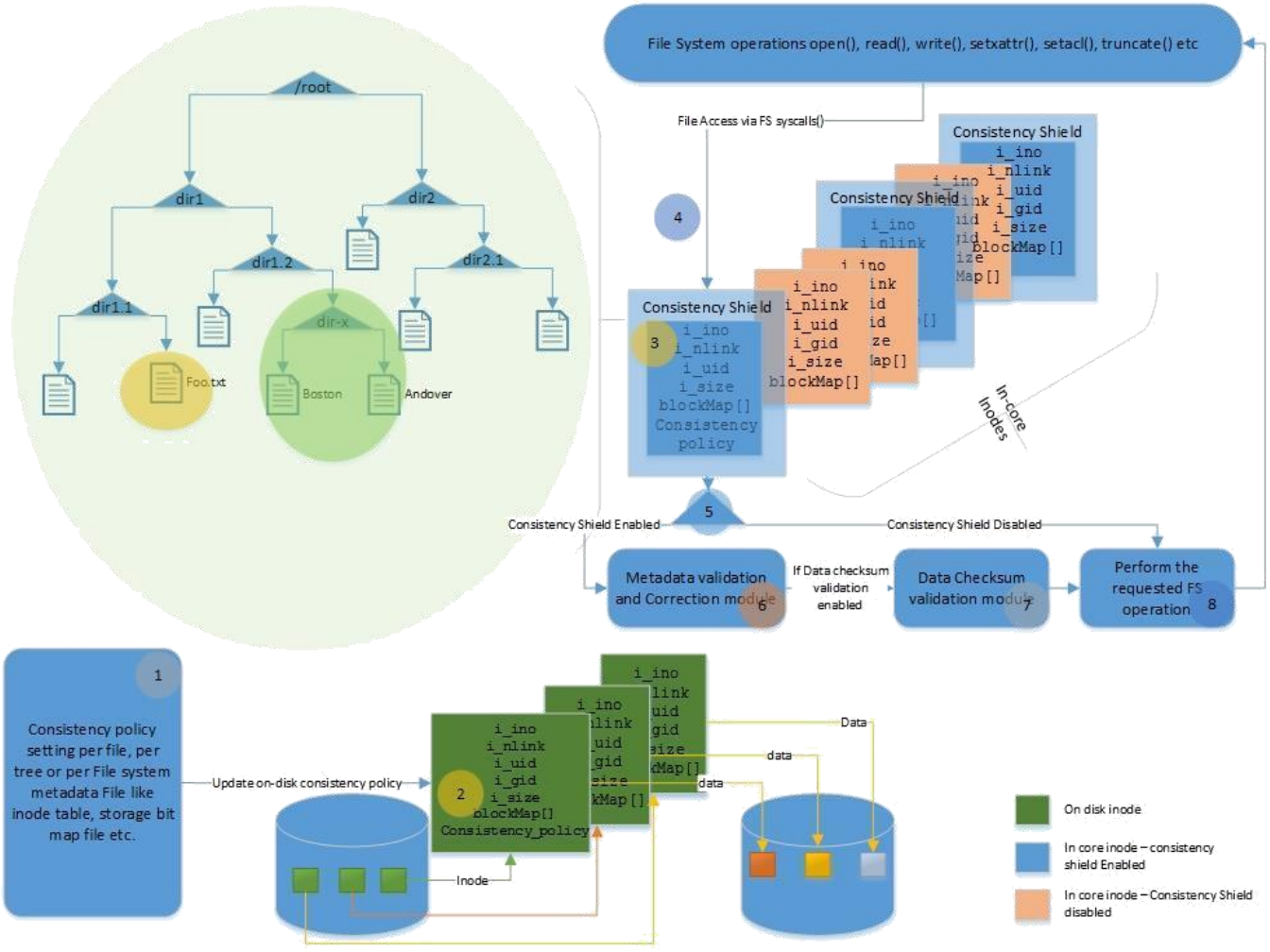
```
Procedure Object_Access (input: File in-core inode or File path or on-disk inode
number)
Begin
    Consistency Policy = read in-core or on-disk
    object consistency policy if (Consistency Policy
    set) then // Figure 1 (stage 4 to 8)
    Begin
        Invoke procedure
    Check_object_Consistency_based_on_Policy (inode) else
    if (Consistency Policy not set) then // Figure 1 (stage 4
    and then to stage 8)
    Begin
        Execute the user requested File system operation and return.
    End
End
```

Storage software applications may purely rely on a filesystem to assure file/directory level consistency all the time. These application's metadata files are highly crucial for the application to work which are tightly coupled with user data. Therefore, it's even more important to maintain all time and online data consistency to respective application's metadata files. So, we propose a mechanism for the application developers to define set of files and directories that are critical for the application to work consistently. Given the information, filesystem can then selectively check for online consistency on the application mandated files and directory trees before allowing access to the application.

Advantages

- Our method does on demand and selective files/directory/File system metadata objects consistency check as oppose to entire filesystem check.
- Unlike traditional filesystem check, our proposal is to perform consistency validation inline as the object gets accessed with respect to user specified files/directory tree level.
- File System metadata objects consistency shall be selectively ON based on a policy definition guarantees application data consistency all time.

Figure 1: Policy based per object consistency shield architecture diagram



Disclosed by Anand A Ganjihal, Jayasankar Nallasamy and Hiro Rameshlal Lalwani, Hewlett Packard Enterprise