High level design of End-to-end data integrity in Mero

By Madhavrao Vemuri <madhav_vemuri@seagate.com>
Date: 2017-07-03
Revision: 2:0

[Text in square brackets and with a light-green background is a commentary explaining the structure of a design document.]

This document presents a high level design (HLD) of the Mero End-to-end data integrity. The main purposes of this document are: (i) to be inspected by the Mero architects and peer designers to ascertain that high level design is aligned with Mero architecture and other designs, and contains no defects, (ii) to be a source of material for Active Reviews of Intermediate Design (ARID) and detailed level design (DLD) of the same component, (iii) to serve as a design reference document.

The intended audience of this document consists of Mero customers, architects, designers and developers.

High level design of End-to-end data integrity in Mero

Introduction

This document discusses a high level design for end-to-end data integrity in Mero. The data integrity mechanism is designed to detect and inform users about data integrity failures.

Definitions

DI Data integrity

Requirements

[r.di] This feature will be provided for user-data. Checksum is computed for each block of data and it's end-end integrity is checked.

[r.di.unit-size] Data for each target is divided into blocks of unit size and CRC is computed for each block.

[r.di.configure] It shall be possible to configure the data-integrity feature at various levels of granularity. Checksum algorithm to be used for each block is also selected based on configuration.

Design Highlights

The primary goal of this design is to provide end-to-end data integrity for the user data. Mero ObjectStore is a distributed storage system. The data integrity feature detects errors during network transfer as well as storage and reports the same to user and Halon. DI is supported for AD stob only.

Following levels of granularity will be supported based on configuration

1. Enable DI at m0t1fs/clovis and ioservice
2. Enable DI at m0t1fs/clovis only
3. Enable DI at ioservice only
4. Disable DI (default)

Also above levels are supported on per pool basis.

Clovis also needs to support DI like m0t1fs.

Functional Specification

User data is divided into blocks as per unit size of the file and checksum is computed at m0t1fs and will be sent to ioservice either through iofop or RPC-AT api's. Checksum is computed and/or verified at m0t1fs/clovis or at ioservice depending on operation type. AD stob needs to be modified to store checksum values.

Logical Specification

During **write** operation m0t1fs/clovis will send DI data to ioservice onwire and based on configuration DI is verified at ioservice and if successful then written to the AD stob. Otherwise it will return error to m0t1fs and write won't be done.

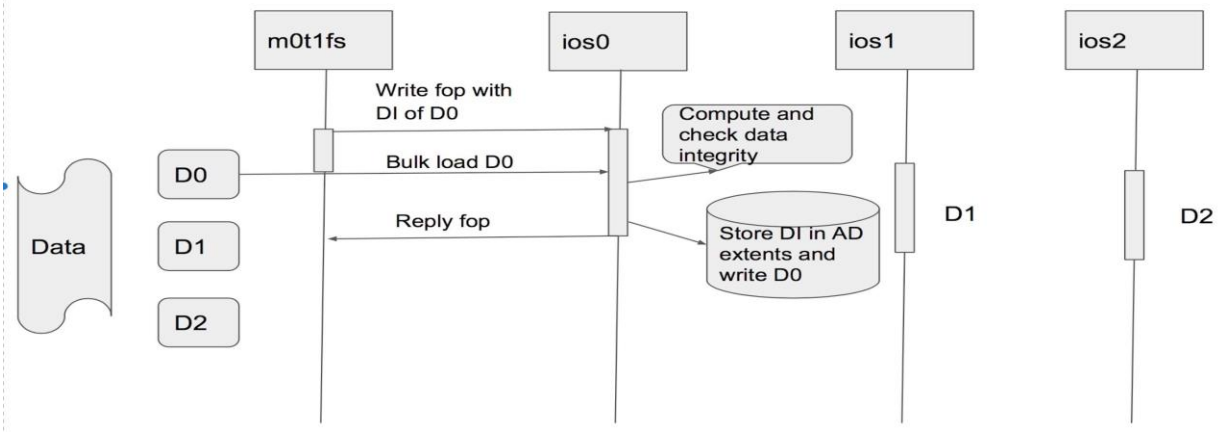
During **read** operation if ioservice detects corruption, then error will be returned to m0t1fs/clovis and disk error is notified to halon. If no data corruption is found then DI data is sent through reply fop or RPC-AT api's. Again at m0t1fs/clovis data integrity is checked. If corruption is detected then error is returned to the user/halon and degraded read is done.

During **SNS repair** operation if corruption is detected during read operation, repair of that group is skipped and disk error is also notified to halon. Also during write checksum values are computed and stored in AD stob in disk.

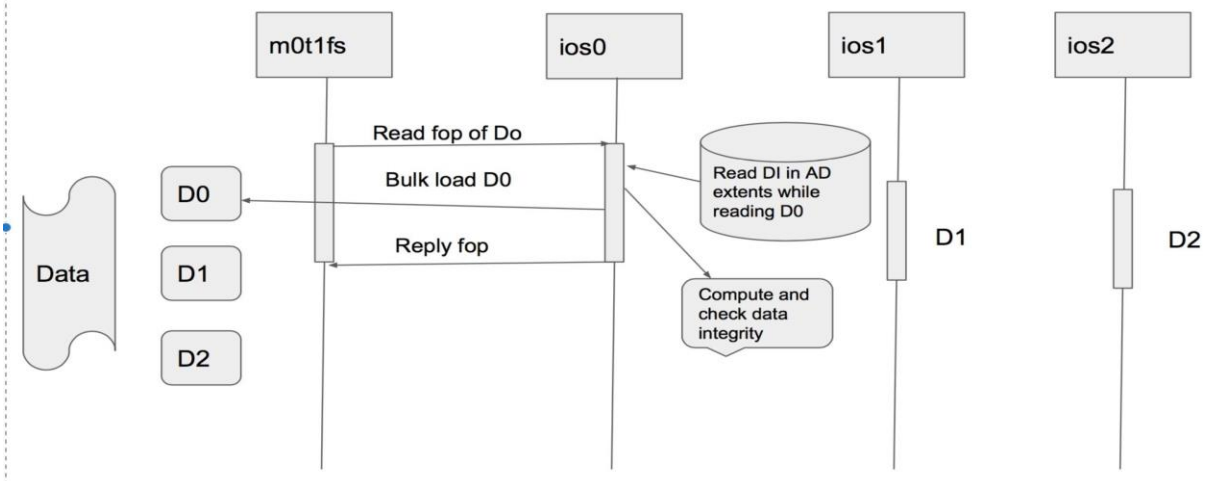
Use cases

Following diagram shows the path of DI during Write, Read and SNS repair,

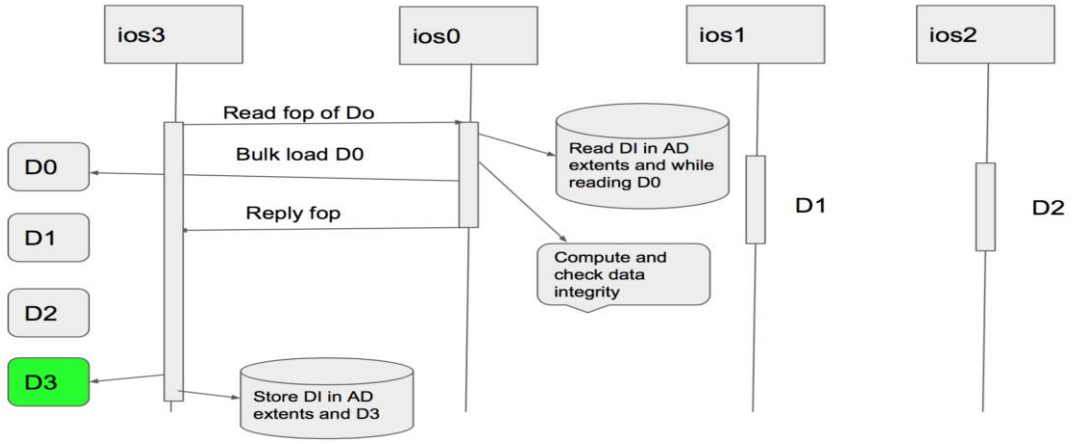
Write



Read



SNS



Analysis

Analyse the performance impact of storing di data for different block sizes on AD stob.

Deployment

Due to changes in AD stob, new version of mero may not be compatible with older versions.

Test cases

1) Check DI with following configuration options

	m0t1fs/clovis	ioservice	AD	
Write	DI compute	DI check	DI store	
	DI compute	-	DI store	
	-	-	-	Default
Read	DI check	DI check		
	Di check	-		
	-	-	-	Default

2) Test with different unit sizes of data.

3) Test with RMW

References

1. High level design of End-to-end data integrity in Mero