A Method to Handle IOs between two arrays with different Performance Capabilities

Dr. M. K. Jibbe
Director of Quality Architect Team, NetApp APG
mahmoudj@netapp.com

Bernard Chan,
Quality Architect, NetApp APG
Bernard.chan@netapp.com
Abstract

- The method of this paper is to resolve the issue associated with I/O access to arrays with different performance capabilities especially in Remote Volume Mirroring (RVM) activities. The issue is resolved by negotiating for each storage array’s capability up front and based on the information exchanged, create an ‘Inter-array IO flood gate’ at the primary storage array and set a higher priority for IOs initiated by a storage array. When the remote storage array performance capability changes, the remote storage array will inform the primary storage array of the change. The primary storage array can then query the remote storage array for the new capability.
Basic Steps

- Here is how the method works in a three step approach.
  - Step 1: Determining the Inter-array IO queue size
  - Step 2: Setting the Inter-array IO queue size
  - Step 3: Executing based on the Inter-array IO queue size
Background

- RAID Storage controller functions as both Initiator and Target in Remote Mirroring environment
- Types of Remote Mirroring – synchronous (sync) & asynchronous (async)
- Sync Remote Mirror – controller waits for response from remote controller before returning status to host
Basic RVM Setup

Primary Site

Host

Primary Zone - \( Z_p \)

Fabric Switch

Mirror Zone - \( Z_m \)

Secondary Zone - \( Z_s \)

Secondary Site

FC

CTRL A – CTRL A’
CTRL B – CTRL B’

HIC Slot 4
Storage Array

Drive Trays

1-2 Kms FC Cable

10-50 Kms FC Cable

Ethernet Switch
Challenge – Sync Remote Mirror

IOs can be held up on Primary Controller if Remote Controller
- is a slower controller
- encounters a hiccup
- is busy as it gets “pounded” by local hosts
Factors affecting a storage array’s performance include:

- Volume count
- Amount and type of IOs
- Type of drives (eg. SAS, SATA, FC or SSD)
- Host and drive channel speed
- Background activities (eg. Data Scrubbing, Dynamic Segment Sizing, Volume Copy, etc.)
Implication

- IO respond time can get choppy
- Host IO timeout → IO error
- IO Aborts
- Mirror loss of sync
Existing Solution

- Increase Command Timeout value
- Have Primary and Remote storage of equal performance capabilities and same configuration
- Reduce the queue depth on the initiator
Downside of Existing Solution

- Increase Command Timeout value
  - Host application can’t tolerate the higher timeout value
  - Some IO driver parameters are not tunable
- Have Primary and Remote storage of equal performance capabilities and same configuration
  - Inflexible storage requirement
- Reduce the queue depth on the initiator
  - Impossible to determine what this queue depth is – no ‘magic’ number
  - Adding an additional host invalidates this value
Proposed Solution – overview

- Negotiate for each storage array’s capability during initialization – Inter-array IO flood gate
- Notify remote array if any changes on array capability
- Inter-array IO flood gate controls the amount of IOs to be sent to remote array
- If queue is full, delay loss and return Vendor Unique Check Condition
Proposed Solution – 3 step approach

1. Determining the Inter-array IO queue size
2. Setting the Inter-array IO queue size
3. Executing based on the Inter-array IO queue size
Step 1 Determine Inter-array IO queue size

- Baseline Inter-array IO queue size for each storage array type is defined in NVSRAM.
- Controller reads from NVSRAM this queue size during Start-of-Day.
- Initial multiplier value is set to 1.
- Multiplier is reduced when storage array performance degrades due to certain event.
- Inform the primary storage of multiplier change if there’s an inter-array communication established.
Step 2 Setting Inter-array IO queue size

- Primary storage array queries remote storage array for Inter-array queue size in 4 scenarios
  1. Primary array Start-of-Day
  2. Setting up the Inter-array communication
  3. Inter-array link is broken and re-established
  4. Remote array informs primary array of change in multiplier
Step 3 Executing based on Inter-array IO queue size

- Multiplier is applied to the size of the Inter-Array Queue Size.
- IOs are sent to remote array if queue is not full. IOs will also be tagged with a higher priority tag.
- If queue is full, delay for a brief period and return the IO back to the host with a Check Condition – allow Remote controller to catch up.
Example

- User configures two storage arrays with different ‘horsepower’ (Primary storage > Remote storage)
- Baseline queue size is set at 512 (Primary storage) and 256 (Remote storage) respectively
- With data scrubbing running on Remote Storage, multiplier is set at 0.85 → inter-array queue size is now at 217 (256 x 0.85).
- When inter-array comm. is established, Primary queries Remote array.
- Set Primary inter-array queue size to Remote array’s queue size
Example - continue

- If inter-array queue is full, Primary storage will delay the commands for a moment before returning a Busy status to initiator – allow Remote controller to catch up

- When data scrubbing completes, multiplier returns to 1.00 → queue size goes back to 256 (256 x 1)

- Remote array informs Primary array of the multiplier change

- Primary array sets the inter-array queue size back to 256
Advantage

- Resolves issue of host experiencing IO error due to command timing out caused by inter-array pipe being ‘clogged’
- Transparent to the user.
  - No user intervention needed
  - No initiator parameter changes needed
- Allows user to use a slower storage array as the Remote array
Thank you