B-010

RAID システム内蔵型 NAS(3) - 多世代スナップショット機能の高速化-

Embedded NAS for RAID System(3) –Improvement of Multiple Generation Snapshot Function–

須藤 敦之† 山崎 康雄†
Atsushi Sutoh Yasuo Yamasaki

1. はじめに

ファイルシステムの瞬間イメージを維持・提供するスナップショット機能は，オンラインで参照可能な一時的なバックアップを実現する NAS の特徴機能のひとつである．スナップショットの多世代化は，バックアップ頻度の増加を可能にするため，信頼性を向上させる．

RAID システム内蔵型 NAS は，大容量，高信頼，高可用性を特徴とする大型 RAID の NAS 拡張である．そのスナップショット機能として，多世代化および TB クラスのボリュームサイズ対応などを行ってきた．これらの対応のもと，Linux LVM(Logical Volume Manager)のスナップショット機能が抱えるボリュームへの WRITE 性能の問題点について明らかにし，その改善方式について検討と評価を行った．

2. 基本機能

RAID システム内蔵型 NAS の多世代スナップショット機能は，Linux の LVM を変更したものである[1]．最大世代数を固定し，全世代のボリューム上のデータ配置を管理するマッピングテーブルを使用することで，多世代かつ大容量ボリュームに対応したスナップショット機能を実現する．

多世代スナップショット機能の概要について示す(図 1)．スナップショットを作成する対象である運用ボリュームと，スナップショット情報を保持する差分ボリュームを用意する．ここでスナップショットを作成すると，それ以降運用ボリュームの変更(①)を差分ボリュームに退避し(②)，その後で運用ボリュームの更新を反映する(③)．これら一連の処理を，CoW(Copy-on-Write)処理と呼ぶ．

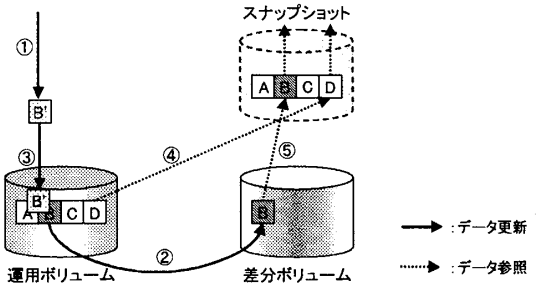

図 1 スナップショットの処理概要

スナップショットを参照する際は，マッピングテーブルに従って運用ボリューム上のデータ(④)と差分ボリュームに退避されたデータ(⑤)とを組み合わせることで，ボリュームイメージを作成する．

3. 従来の CoW 方式と課題

Linux LVM の CoW 処理方式とその課題について示す(図 2)．スナップショット機能は，Linux の LVM ドライバで実現している．対象である運用ボリューム，差分ボリューム，そしてスナップショットボリュームはいずれも論理ボリュームである．

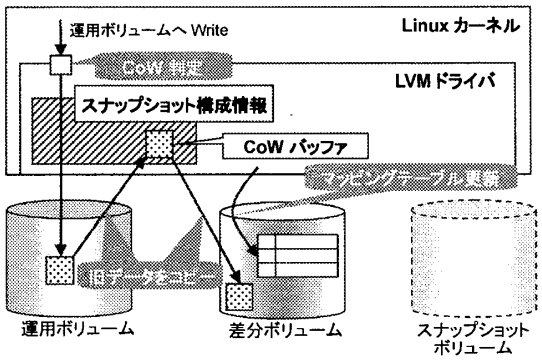

図 2 CoW 処理方式概要

スナップショット作成後，CoW 処理が必要か否かの判定は，運用ボリュームへの書き込み時に LVM ドライバが行う．CoW 処理を行う条件は，(a) スナップショットを作成済みか，(b) CoW 処理を未実行のブロックであるか，の 2 つである．これらの条件を満たすブロックは，以下の処理を行う．

1) 運用ボリュームから以前のデータを読み込み
2) そのデータを差分ボリュームに書き込み
3) マッピングテーブルの更新

マッピングテーブルは差分ボリューム上にあり，差分ボリューム上のデータがどのスナップショットの，何番のブロックに対応するかを保存する．この順に処理を行うことで，障害などによるリセット時にも，スナップショットボリュームの論理的な一貫性を保つようになっている．

このように，スナップショットを作成した場合，CoW 処理によって WRITE 時の I/O 容量がおよそ 3 倍に増加す

† (株)日立製作所 中央研究所
Central Research Laboratory, Hitachi Ltd.

る．そのため，スナップショットを作成していない場合と比較して運用ボリュームの WRITE 性能が悪化する．

Linux の LVM ドライバでは運用ボリューム 1 個につき，CoW 処理を行うための CoW バッファを 1 個保持している．そのため，WRITE が連続すると 1 個ずつしか CoW 処理できず，後続の WRITE が待たされることになる．その結果，スナップショットを作成していない場合と比較して，著しく WRITE 性能が低下していた．

4. CoW 処理並列化方式

4.1 方針

CoW バッファの数を増やし，CoW 処理を並列動作可能にすることで運用ボリュームの WRITE 性能を向上させる方式を考案した．Linux LVM では，運用ボリュームと差分ボリュームの 1 組に対して 1 個の CoW バッファを使用していた．このバッファを保持する方法を変更し，LVM ドライバ内で多数のバッファをプールし，その中から各運用ボリュームの CoW 処理で必要なときに割り当てる方式とした(図 3)．

CoW バッファを運用ボリュームと差分ボリュームのペアごとに複数割り当てる方式も考えられる．しかし，ボリュームごとにバッファを割り当てた場合，

1) メモリ使用量がボリューム数に比例して増加する

2) I/O 性能を超えた CoW 処理は実行できない

などの問題が考えられる．そのため，全ての CoW バッファを一括管理するほうが効率的であると考えた．

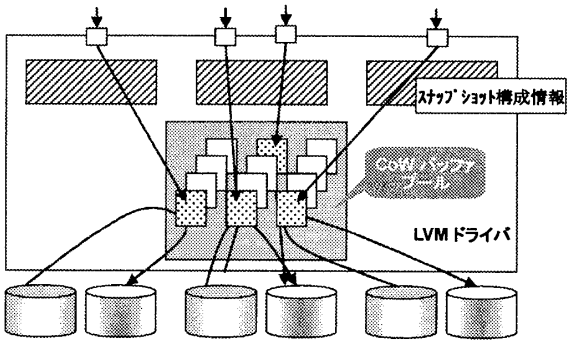

図 3 CoW 処理並列化方式概要

4.2 CoW バッファの排他方式

従来の方式では，運用ボリュームと CoW バッファは常に 1 対 1 に対応していた．そのため，CoW バッファの使用中は，その運用ボリュームへの WRITE を待たせることで，CoW バッファの 2 重使用や，あるブロックの CoW 処理の 2 重実行は起こらなかった．

しかし，本方式では複数の CoW 処理を同時実行するため，新たな CoW バッファの排他方式を作成する必要があった．そこで，CoW バッファごとに新たな情報を付加し，その情報を使用することで排他方式を実現した．その情報は、(1) CoW 実行中の運用ボリューム ID，(2) CoW 実行中のブロック ID の 2 種類である．また，CoW バッファを使用中でない場合は，これらの情報に初期値を代入してクリアしておく．

この情報を使用して，CoW バッファおよび CoW 処理の排他は，以下のように実現できる．まず，CoW 処理が必要なブロックへの WRITE があった場合，CoW バッファプールをその運用ボリューム ID とブロック ID とで検索する．そして

(1) 既に CoW バッファを実行中であれば，その WRITE は WAIT

(2) 未実行であれば，使用可能な CoW バッファを確保し，CoW 処理を開始

という処理を行う．

5. 評価

CoW 処理並列化方式の適用前と適用した試作版とで，スナップショット作成後の運用ボリュームへの WRITE 性能を測定し比較した (図 4)．

測定は，1 個の運用ボリュームへ複数のプロセスで同時に WRITE を実行した．そして，そのプロセスが完了するまでの時間と WRITE 容量の合計から性能を求めた．

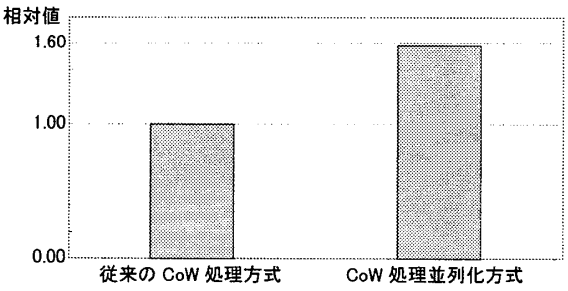

図 4 運用ボリュームへの WRITE 性能比較

図 4 から分かるように，CoW 処理並列化方式を用いることで，1 個の運用ボリュームに対する WRITE 性能は，およそ 60%向上した．

6. おわりに

多数の CoW バッファを LVM ドライバで管理し，CoW 処理を並列に実行する本方式を用いることで，スナップショット作成時に低下していた運用ボリュームへの WRITE 性能の高速化を実現した．

参考文献

[1] 中野隆裕，山崎康雄，藤井直大：RAID システム内蔵型 NAS(2) –多世代スナップショット機能–，情報処理学会第 66 回全国大会，5D-3(2004)