連鎖ネットワークRAIDの実装方式の検討

市川俊一[†]　豊田真智子[†]　高橋克巳[†]

1. はじめに

計算機によって扱われるデータ量は年々増加し，ストレージデバイスの設置・運用コストが増大している．多くの企業では，計算機とストレージデバイスがシステムバスで接続される従来のストレージアーキテクチャDirect Attached Storage (DAS) から，計算機とストレージデバイス間を高速なネットワークで接続する Storage Area Network (SAN) とよばれるストレージアーキテクチャへ移行している．また，PC のプロセッサ性能の向上と低価格化を背景に，高性能計算機の分野では PC を Ethernet などの高速なネットワークで結合した PC クラスタ・PC グリッドが普及している．しかし，PC を主体としたストレージクラスタを構成し SAN として用いる場合，PC の信頼性の低さがストレージデバイスの性質として特に問題となる．そこで，本研究では PC を用いて SAN を構成することを想定し，PC に保存されるデータの信頼性を向上させる手法を検討する．

2. 既存手法

ディスクドライブを高信頼化する手法として RAID が広く用いられている．しかし，ストレージクラスタの規模が大きなりノード数が多くなると，RAID では十分な信頼性を確保できなくなる．そこで，Qin ら[1] は障害発生時のアレイの再構築にかかる時間を短くすることで，データの信頼性を高める手法を提案した．しかし，ネットワークにおける障害など，複数のノードで同時に発生する障害には対応できないという問題がある．

Intermemory[2] は消失訂正符号を使ってブロックレベルのストレージを提供する分散型のシステムである．IM-0 というプロトタイプ実装は，データを 16 out of 32 符号で分配することで高い信頼性を実現する．また，各ワークステーションでデーモンが動作し，自律的に動作することで中央の制御を不要にしている．しかし，提供できるストレージは write-once 型に限られ，ISO-9660 CD-ROM イメージとしてマウントしなければならない点が問題である．

3. 検討手法

RAID ではグループに割り当てられるノードの数が少数で固定であるため，同時に発生する障害に弱いという問題があった．そこで，筆者らは連鎖ネットワーク RAID[3] というデータ配置モデルを提案し，それが消失訂正符号並みの高い信頼性を実現することを検証した．本稿では，そのモデルを用いたシステムの実装方式について検討する．

3.1 ボリューム操作

連鎖ネットワーク RAID は，ノードが保持する物理ボリュームから仮想的に高信頼な論理ボリュームを提供する．通常の RAID では，1 つの論理ボリュームは少数の物理ボリュームから構成され，それらは入れ子になる場合はあるが，互いに独立のグループとなる．連鎖ネットワーク RAID では，論理ボリュームと物理ボリュームは多対多の関係を持ち，全体で一つのグループを構成する．連鎖ネットワーク RAID は次の 6 つのコマンドで，ボリュームの構成を操作する．

- 物理ボリュームの追加 (add physical)
- 物理ボリュームの削除 (remove physical)
- 論理ボリュームを物理ボリュームを選び定義 (add logical as physical)
- 論理ボリュームの削除 (remove logical)
- 物理ボリュームに複数の論理ボリュームを結びつけ (bind physical with logicals)
- 物理ボリュームから複数の論理ボリュームを解放 (unbind physical with logicals)

ここで，複数の論理ボリュームが結びつけられた物理ボリュームは，それら論理ボリュームの XOR 演算結果を保持する．また，論理ボリュームは最低でも 1 つの物理ボリュームと結びつけられる．

3.2 物理ボリュームの状態

物理ボリュームが bind と unbind 操作で論理ボ

† 日本電信電話株式会社 NTT 情報流通プラットフォーム研究所
NTT Information Sharing Platform Laboratories, NTT Corporation

リュームとの結びつけが変更になった時と障害や一時的な停止から復旧した時に，それが保持するデータは初期化・最新化されなければならない．この同期処理は，論理ボリュームへの読み書き要求を停止することなく，実現する必要がある．そこで，物理ボリュームの状態は次の 3 つのいづれかで管理される．

- 最新化されており，読み書き可能 (UPDATE)
- 同期処理中であり，書き込みのみ (SYNC)
- 最新化されておらず，利用不可 (TAINT)

同期処理中という状態を設けることで，結びついた論理ボリュームへの処理を停止する必要がなくなり，自由に bind と unbind 操作を実行できる．

3.3 要求の解決処理

論理ボリュームへの読み出し要求は，物理ボリュームへの読み出し要求の形に変換される．この読み出し解決処理では，その状態が UPDATE である物理ボリュームが対象となる．ある物理ボリュームに注目すると，その物理ボリュームに結びついた論理ボリュームのうち，1 つの論理ボリュームを除いてデータが取得可能であるとき，その論理ボリュームはその物理ボリュームを用いることでデータが取得可能になる．この規則を利用して，すべての物理ボリュームを繰り返し評価することで，論理ボリュームへの読み出し要求にどの物理ボリュームを用いればよいかが明らかになる．

論理ボリュームへの書き込み要求は，物理ボリュームへの読み出し要求と書き込み要求の形に変換される．この書き込み解決処理では，その状態が UPDATE と SYNC である物理ボリュームが対象となる．論理ボリュームに結びついたすべての物理ボリュームが書き込みの対象となる．各物理ボリュームについて書き込むデータを算出するために，結びついたすべての論理ボリュームに対して読み出し解決処理が行われ，それらの物理ボリュームが読み出しの対象となる．

3.4 排 他 制 御

1 つの物理ボリュームに複数の論理ボリュームが結びついているため，異なる論理ボリュームへの要求が同じ物理ボリュームへの要求に変換される．この時，これらの処理を同時に行ってしまうと，データの整合性が失われる．そのため，物理ボリュームへの要求が競合する論理ボリュームへの要求は，排他制御を行い同時に実行されないようにしなければならない．各物理ボリュームは読み書きロックを持ち，要求の処理は必要な物理ボリュームのそれらのロックを事前に取得する．

3.5 システム構成

これまでに述べた処理を各ノード上で自律分散的に行うことを考えると，ネットワークの遅延による性能の劣化と異常系処理の複雑化が問題になる．しかし，単純にすべての処理を単一のノードで行うと，そのノードの処理性能がボトルネックとなり，スケーラビリティが低くになる．そこで，制御情報とブロック I/O を分離して，スケーラビリティの向上を図る．システムは次のコンポーネントで構成される．

- initiator_block: 論理ボリュームへ要求を出す
- target_block: 物理ボリュームを提供する
- map_handler: ボリューム情報の保持と操作，要求の解決処理と排他制御を行う
- dispatcher: initiator_block から要求を受け，map_handler と制御情報をやり取りして，ブロック I/O の処理を行う
- synchronizer: map_handler から同期処理のための要求を受け，ブロック I/O の処理を行う

また，map_handler と dispatcher との間の処理は，次の 4 つのフェーズから成る．

- Resolve: dispatcher が論理ボリュームへの要求を伝える
- Grant: map_handler が解決処理で変換し，ロック取得をした物理ボリュームへの要求を伝える
- Release: dispatcher が物理ボリュームへの要求の処理結果を伝える
- Ack: map_handler が論理ボリュームへの要求の処理結果を伝える

Resolve に対してすぐに Ack が返る場合や，Release に対して再度異なる Grant が返る場合もある．

4. 今後の予定

プロトタイプを用い PC 上で性能の評価を行う．

参 考 文 献

1) Qin, X. et al.: Evaluation of Distributed Recovery in Large-Scale Storage Systems, *HPDC-13 '04, 13th IEEE International Symposium on High Performance Distributed Computing*, pp. 172–181 (2004).
2) Chen, Y. et al.: A prototype implementation of archival intermemory, *In Proceedings of the 4th ACM Conference on Digital Libraries*, pp. 28–37 (1999).
3) 市川俊一, 高橋克巳: 高信頼，高可用な分散ストレージを実現する連鎖ネットワーク RAID, SACSIS2005, IPSJ Symposium Series Vol.2005, No.5, pp. 99–106 (2005).