Vol.2014-ARC-210 No.17
Vol.2014-OS-129 No.17
2014/5/15

分散ブロックストレージの広域化

福本 佳史[†]　三嶽 仁[†]　小西 隆介[†]　湯口 徹[†]

概要：仮想化環境において，拠点単位の大規模障害が起こった場合でもデータ損失回避と短時間での情報システム復旧を可能とするため，離れた 2 拠点間で仮想マシンが利用するブロックストレージを常に同期させる技術が注目されている．拠点間でのブロックストレージの同期は商用製品や DRBD を利用することで実現可能だが，導入コストが大きく適用可能なアプリケーションが限られ，スケール性や構成の柔軟性に欠けるなどの問題がある．本研究は広域ネットワークをまたいだ 2 拠点での運用を想定し，かつ低コストに構築可能で柔軟にスケールする分散ブロックストレージの実現を目標としており，スケーラビリティを備えた分散ブロックストレージである Sheepdog をベースとして広域化に取り組んでいる．拠点単位の大規模障害時におけるデータ損失回避と，拠点間の広域ネットワークのトラフィック量低減のために，Sheepdog のデータ配置・送受信部のアルゴリズムの改良を行い，距離の離れた 2 つの拠点を模擬した環境において Active-Standby 構成でのデータベースのフェイルオーバーが可能であることを確認した．

1. はじめに

近年，クラウドコンピューティングの普及によって情報システムを仮想化する事例が増加している．仮想化環境の構築において，仮想マシンに仮想ディスクを提供するため，共有ストレージ製品を導入することが主流となっている．その理由として，ストレージ仮想化機能を用いて容量効率や運用性を高めることができる点，仮想マシンのライブマイグレーションが可能となる点などが挙げられる．しかし，そのような一般的に普及している共有ストレージ製品を利用する場合，システムの負荷やデータの増減に合わせてストレージの性能・容量の拡張や縮退を行う際に，製品仕様や初期設計による制約やコスト増大などの問題がある．そのため，共有ストレージ製品の代替として，初期構築以降も物理リソースの増減設によって柔軟に性能・容量をスケールさせることができる VMware Virtual SAN[5] や Ceph[2] を例とした分散ブロックストレージを利用する例も登場している．Sheepdog[1] は分散ブロックストレージを実現するためのソフトウェアのひとつであり，仮想化環境において十分な仮想化機能・性能・スケーラビリティを備えたストレージを，複数台の安価な PC サーバを用いて低コストに構築することが可能である．

仮想環境におけるストレージにスケーラビリティが求められる一方で，激甚災害や停電などを原因とした大規模障害が発生した場合でも，データ損失回避・短時間での復旧が可能な仮想化環境への要求が高まっており，それに寄与するストレージも求められている．広範囲に及ぶ障害が発生した場合，単一の拠点内で物理リソースやデータを冗長化していてもその全てが同時に停止してしまう可能性があり，さらには拠点内の物理リソースへの電源供給が長時間に渡って途絶えることも懸念される．そのような事態が起こった場合でも，ある拠点内のブロックストレージの複製を，例えば電力会社の管轄地域をまたいだ遠隔地にある第 2 拠点に常に保持させることによって，データの損失を回避し，障害を免れた拠点で障害前の情報システムを素早く再開することが可能になる．そのような複数拠点間のブロックストレージの遠隔複製に利用可能な製品として VPLEX Metro[4]，オープンソースソフトウェアとして DRBD[3] などが注目されている．しかし，特に前者は導入に要するコストが非常に高く初期構築後の構成変更にも追加的なコストがかかり，どちらも製品仕様や物理リソースの制約など柔軟な拡張・縮退を阻害する問題がある．

本研究では，以上に挙げた仮想化環境のストレージに対する要求である柔軟なスケール性と，大規模障害に対するデータ保護・素早い復旧を満たす分散ブロックストレージ

[†] 日本電信電話株式会社
NTT ソフトウェアイノベーションセンタ
{fukumoto.yoshifumi,mitake.hitoshi,konishi.ryusuke,
yuguchi.toru}@lab.ntt.co.jp

技術の確立を目標としている．本論文では，分散ブロックストレージ Sheepdog をベースとして，複数拠点を用いたデータ保護と遠隔拠点間のネットワーク通信量削減を行う機能を追加した広域化版の Sheepdog について述べる．実装のベースとした Sheepdog は単一障害点のない対称型のクラスタ構成で動作する分散ブロックストレージであり，クラスタを構成する複数のノード（PC サーバ）間でオブジェクト（分割された仮想ディスクデータ及びそのコピー）を配置アルゴリズムに基いて複数のノードに保存している．ノードの追加・離脱があった場合，自動的かつ少ないデータ移動量でオブジェクトの適切なノードへの再配置・冗長度回復が行われるようになっており，さらにクラスタを構成する任意の PC サーバから仮想ディスクにアクセス可能であるため，ストレージ容量や多数のアクセスに対する性能の拡縮が容易である．広域化版 Sheepdog ではノードに対して所属拠点 ID を付与し，可能な場合は必ず複数の拠点にオブジェクトを配置することで，拠点単位の障害の際も残った拠点で仮想ディスク全体を復元可能としている．また，仮想ディスクからの読み込み時には対象となるオブジェクトを保存しているノードをランダムに選んでオブジェクトを取得する方式に同一拠点内ノードを優先する処理を加えており，仮想ディスクへの書き込みアクセス時において遠隔拠点の複数オブジェクトに書き込みを行う場合は複製方式を適切に切り替えることによって，距離による遅延が大きいと想定される拠点間のネットワークトラフィックを削減している．

ネットワークエミュレータを用いて距離の離れた 2 つの拠点を再現し，従来の Sheepdog と実装した広域化版 Sheepdog で 2 拠点にまたがる一つの分散ブロックストレージクラスタを構築して，各 Sheepdog から提供される仮想ディスクの性能評価と拠点単位の障害時のデータ保護・復旧の評価を行った．広域化版 Sheepdog の仮想ディスクは，特に書き込み性能において拠点間ネットワークの距離が大きいほどネットワーク遅延が支配的なボトルネックとなり，読み込み性能において拠点間ネットワークの遅延の影響を回避できることを確認した．また，拠点単位の大規模障害を再現した場合，オブジェクトの複製数以上のノードが同時に停止するため，従来の Sheepdog では仮想ディスクが破損して I/O が停止してしまうのに対し，広域化版 Sheepdog は残った拠点において仮想ディスクへの I/O を継続可能で，障害直前までのトランザクションが維持されたデータベースのフェイルオーバーが可能であることを確認した．

2. 遠隔複製

地理的な広範囲に及ぶ大規模障害が起こった場合でも情報システムのデータ保護・短時間でのシステム復旧を実現するため，遠隔地の拠点にデータの複製を保持することを遠隔複製と呼ぶ．遠隔複製は情報システム内のアプリケーション・ストレージのどちらかのレイヤで制御され，大別して 3 つの複製方式が存在する[*1][15]．

アプリケーションレイヤで遠隔複製を制御する場合は，静止点をアプリケーション側で保証することが可能なため，複製データからのシステムの復旧を確実に行うことができるメリットがある一方で，遠隔複製に対応したアプリケーションのみが保護対象となる適用範囲の狭さと，複製処理に仮想マシンの CPU リソースを消費してしまうというデメリットがある．ストレージレイヤで遠隔複製を制御する場合は，複数のアプリケーションをまとめて保護対象にしやすく，複製に仮想マシンの CPU リソースを消費しないというメリットがある一方で，アプリケーションの静止点を保証できず破損した仮想マシンを復旧してしまう可能性があるというデメリットがある．

スケジュール方式は，1 日や 1 週間といった特定の間隔で遠隔拠点に複製を送る方式である．この方式ではアプリケーションの静止点保証がしやすく，復旧時にどの程度前の時点までデータが巻き戻るか（RPO）を予測可能で，複製間隔の調整によって遠隔複製に必要な拠点間のネットワーク帯域幅を低減できる．しかし，他の方式よりも比較的複製間隔が長いため複製を作成してから障害が発生するまでに失われるデータ量が大きい．

非同期方式はキャッシュを用いてデータの読み書きを一定量保持しておき，キャッシュの内容を常に遠隔拠点に送る方式である．障害が発生した際はキャッシュ内のデータのうち遠隔拠点に未送信の部分はデータが失われるが，RPO をスケジュール方式よりも小さくすることが可能で，キャッシュに書き込んだ段階でアプリケーションに書き込み完了を返答するため通常の I/O 性能への影響を抑えることが可能である．一方，常に複製を作成し続けるためにアプリケーションの静止点保証が難しく不完全な複製を行ってしまう可能性があり，RPO はキャッシュの負荷状況によって異なるため予測が難しい．

同期方式は遠隔拠点への複製の送信・書込が完了するまで I/O 完了としない方式である．複製は常に最新であることが保証されるため RPO ゼロかそれに限りなく近い値となるが，I/O 毎に遠隔拠点への書込完了を待つため拠点間の長距離ネットワークの遅延時間が直接影響し，通常の I/O 性能が低下する．アプリケーションに対して十分な I/O 性能を提供するために，遠隔複製を行う拠点間の距離を一定以内に抑えて遅延時間の影響を抑制する必要がある．

本論文では，既存の様々なアプリケーションを，データの巻き戻りなく（RPO がゼロ），極力短時間で復旧させる（RTO の短縮）ため，ストレージレイヤでの同期方式による遠隔複製を採用し，分散ブロックストレージである

[*1] http://ascii.jp/elem/000/000/467/467007/

Sheepdog への遠隔複製機能追加に取り組んでいる．

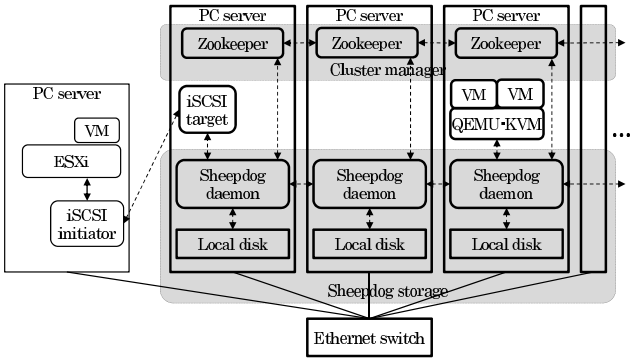

図 1 Sheepdog クラスタの構成

3. 従来の Sheepdog

本論文がベースとしている Sheepdog の全体構成を図 1 に示す．Sheepdog は単一障害点のない対照的なクラスタ構成で動作するようになっており，各ノード（PC サーバ）に内蔵されているローカルディスクから一つの大きな仮想ストレージを形成し，そこから仮想マシンのために任意のサイズの仮想ディスクを切り出して提供することができる．全ての仮想ディスクはクラスタを構成するどのノードからもアクセス可能であり，QEMU[7] のブロックドライバや iSCSI target（tgt）[6] から利用可能である．Sheepdog が提供する仮想ディスクはコピーオンライト・スナップショット・クローンなどの一般的な共有ストレージ製品と同等の仮想化機能を備えている．

従来の Sheepdog は一つの拠点内で運用することを前提として設計されており，広域化版の実装に関連する部分について詳細に説明する．

3.1 ノードリストの管理

Sheepdog ではクラスタを構成するノードリストの管理を Zookeeper[11] または Corosync[12] の仮想同期機能を用いて実現している．ノード追加・離脱があった場合は，ノードリストの変更の通知が 1 ノードずつ保証された順序で各ノードに送られ，各ノードではノードリストの履歴を全てローカルディスクに保存する．ノードリストのバージョン番号（epoch）はデータの一貫性保証に利用されている．

ノードリストの管理に Zookeeper を利用する場合，全ての Sheepdog のノードが Zookeeper のノードを兼ねる必要はなく，Sheepdog クラスタの外部に Zookeeper クラスタを動作させることも可能であるし，Sheepdog ノードのうちの奇数個のノードが Zookeeper ノードを兼ねても良い．Sheepdog デーモンを起動する際に Zookeeper ノードの IP アドレスを一つ以上指定することで，Sheepdog ノードを Zookeeper に管理させることができる．ネットワーク分断が発生した場合は，過半数の Zookeeper ノードにアクセス可能な Sheepdog ノードがクラスタを継続し，小数側の Sheepdog ノードへのアクセスには I/O エラーを返答することで，いわゆるスプリットブレーンを回避し，一貫性を確保している．

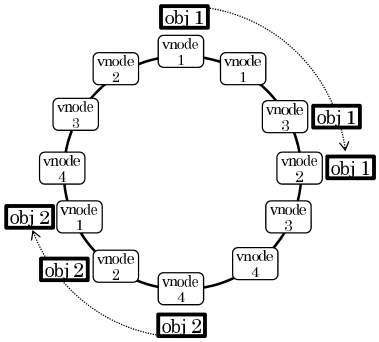

図 2 コンシステントハッシュ法によるオブジェクトの配置

3.2 オブジェクトの配置アルゴリズム

Sheepdog は仮想ディスクをデフォルトでは 4MB のデータに分割し，分割データの複製（オブジェクト）3 つをクラスタを構成する複数のノードに分散して配置することでデータの冗長度を高め，信頼性を確保している．オブジェクトの配置アルゴリズムにはコンシステントハッシュ法 [13] が用いられており，コンシステントハッシュ法のリングは epoch 毎のノードリストから一意に決定され，ノードリストに変更がある度に各ノードにおいて更新されるため，オブジェクトの配置情報を管理するサーバは不要となっている．図 2 はコンシステントハッシュ法のリングを示しており，物理ノードに対応した仮想ノード（vnode）が IP アドレス・ポートのハッシュ値に基いて並べられる．一つの物理ノードに対して複数の仮想ノードが対応しており，仮想ノードの数は物理ノードのローカルディスクの空き容量によって増減する．オブジェクトの配置先はオブジェクト ID から計算したハッシュ値を元に起点となる仮想ノードを決定し，そこからリングを辿るようにして 3 つないしは指定した複製数の仮想ノードを選びとることで決定される．物理ノードの追加や離脱の際は仮想ノードが増減するが，仮想ノードのリング上隣接関係は大幅に変動しないため，新しいオブジェクトの配置先に従って各ノードが自ら持つべきオブジェクトを集めるリカバリ処理の際に，クラスタ全体のオブジェクトの移動量を抑えることができる．

sheepdog デーモンの起動時に物理ノードに zone ID を与えることが可能となっており，例えば同一ラック内の PC サーバや同じ電源を使った PC サーバに同じオブジェクトを集中させたくない場合に，それらの物理ノードに同じ zone ID を与えることによって集中を回避することが可能である．リングを辿りながら配置先を決定してゆく際に，既に配置先として選定した仮想ノードと同じ zone ID

を持つ仮想ノードはスキップすることで実現されており，ユニークな zone ID の数が指定されたオブジェクトの複製数を下回る場合は，ユニークな zone ID の数が実際の複製数（冗長度）となる．

3.3 仮想ディスクへの I/O

Sheepdog はクラスタを構成するどのノードにアクセスしても任意の仮想ディスクに対してデータの読み書きを行うことができる．

データを読み込むために QEMU のブロックドライバや iSCSI ターゲットからアクセスされたノードでは，読み込み対象となるオブジェクトを保持するノードリストをオブジェクトの配置と同じアルゴリズムを用いて決定し，その中に自ノードが含まれる場合はローカルディスクからの読み込みを行い，そうでない場合はランダムに一つのノードを選んでネットワーク越しに読み込み要求を行う．これにより，読み込み要求の負荷分散を実現している．

仮想ディスクへデータを書き込む際は，アクセスされたノードはオブジェクトの配置アルゴリズムで決定された配置先ノードリストに対し，直接かつ並列に書き込み要求を行い，全てのノードの書き込み完了の応答を待ってからクライアントへ書き込み完了の応答を行う direct 方式を採用している．つまり，遠隔ではないが，同期方式で複製の書き込みを行っている．配置先ノードリストへの書き込み要求は並列に行われるので，信頼性向上のために複製数を増やしても書き込み性能が大きく低下することはないが，最も性能が低くレスポンスタイムの大きいノードに律速される．

3.4 オブジェクトのリカバリ

クラスタを構成する各ノードは，Corosync もしくは Zookeeper からのノードリスト変更通知を契機としてリカバリ処理を一斉に開始し，オブジェクトのリバランシング・冗長度の回復を行う．各ノードにおけるリカバリ処理では，まず最新のノードリスト内の全ノードに対して，各ノードが保持する全てのオブジェクトのリストを要求する．その後，重複を排除したオブジェクトリストのうち，最新の epoch において自ノードが配置先となっているオブジェクトに関しては，1 つ前の epoch において配置先となっていたノードにオブジェクトの読み込み要求を送って取り寄せる形でリカバリを行う．その際，1 つ前の epoch において配置先となっていたノードにリカバリすべきオブジェクトが無い場合は，さらに過去の epoch におけるオブジェクト配置先のノードからのリカバリを試みる．

リカバリが完了しておらず，最新の epoch で配置先となるノードにオブジェクトが実際に配置されていない場合，そのオブジェクトに対する書き込み要求は一時保留され，そのオブジェクトのリカバリが優先的に行われる．そして，オブジェクトのリカバリが完了次第，要求された書き込みが行われる．そのため，リカバリ中は仮想ディスクの書き込み性能が低下する可能性がある．

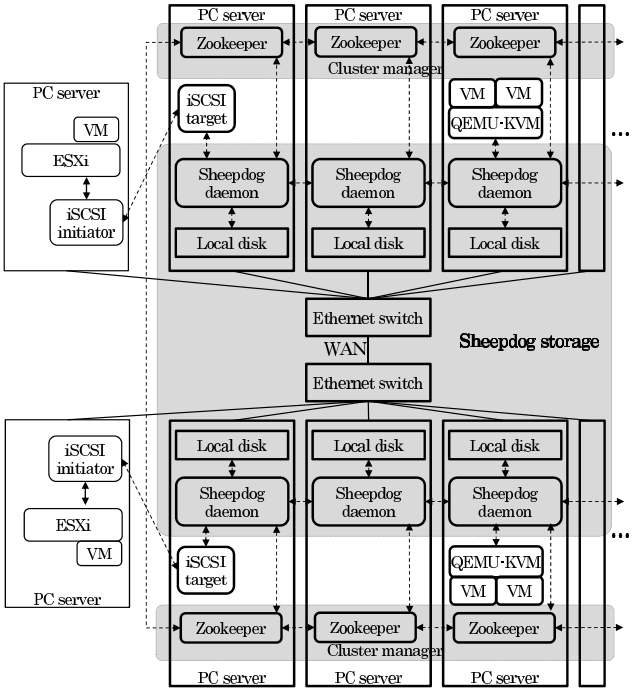

図 3 広域化版 Sheepdog クラスタの構成

4. 広域化版 Sheepdog

従来の Sheepdog は，オブジェクトの複製をハッシュに基いて一様に分散して保存するため，複数の拠点にまたがる Sheepdog クラスタを構築する際に，片方の拠点に全ての複製が集中する可能性がある．そのため，拠点単位の大規模障害が発生した場合に，一部のオブジェクトの複製が全てクラスタから失われてアクセス不可となるため，残った拠点において仮想ディスク全体を復元することができない．また，構成する各ノードは同じネットワークに所属することが前提となっており，帯域・品質・遅延時間等が異なる拠点間ネットワークが混在することが考慮されていない．これらに対応するため，従来の Sheepdog をベースとして広域化版の Sheepdog の実装を行った．広域化版 Sheepdog の全体構成を図 3 に示す．

4.1 拠点を考慮したノードリストの管理

広域化版 Sheepdog では，ノードリストの管理に Zookeeper を利用する場合のみ動作を確認している．

2 拠点にまたがって一つの Sheepdog ストレージを構築しており，片方の拠点が大規模障害によって停止した場合，それがどちらの拠点であっても残された拠点においてクラスタを継続できることが望ましい．しかし，Zookeeper ノードの数は奇数個が推奨されており，どちらかの拠点に

過半数の Zookeeper ノードが存在することになるが，過半数の Zookeeper ノードが存在する側の拠点が大規模障害によって停止してしまった場合は，少数側の拠点ではクラスタの継続ができない．この問題を回避するために，2 つの拠点に同数の Zookeeper ノードを配置し，さらに第 3 の拠点に Zookeeper ノードを置いて，どちらの拠点が停止した場合でも，第 3 拠点の Zookeeper ノードと合わせて過半数となる構成を取る必要がある．

また，2 つの拠点間のネットワーク分断が発生した場合は，一貫性を確保するために，2 拠点のどちらかを優先拠点とし，優先されない側の Sheepdog ノードへの I/O 要求にエラーを応答する必要がある．多数決用の第 3 拠点と 2 拠点の構成では，leader となっている Zookeeper ノードが存在する側の拠点が優先拠点となり，ネットワーク分断時はその拠点でのみクラスタが継続される．Zookeeper を利用する場合はクラスタを構成する全ての Zookeeper ノードの ID と IP アドレスのリストを予め共有する必要があるが，リスト内で最も大きい値の ID を指定した Zookeeper ノードが優先的に leader になるため，優先拠点を制御することが可能である．

Zookeeper ノードは一定間隔（デフォルトでは 2 秒）で互いに通信することで死活監視を行っている．拠点間ネットワークの遅延によって，監視間隔を超える期間通信が途絶えた場合，意図しない Zookeeper ノードの離脱処理が発生する可能性があるため，拠点間のネットワーク遅延時間に合わせて適切に監視間隔を設定する必要がある．

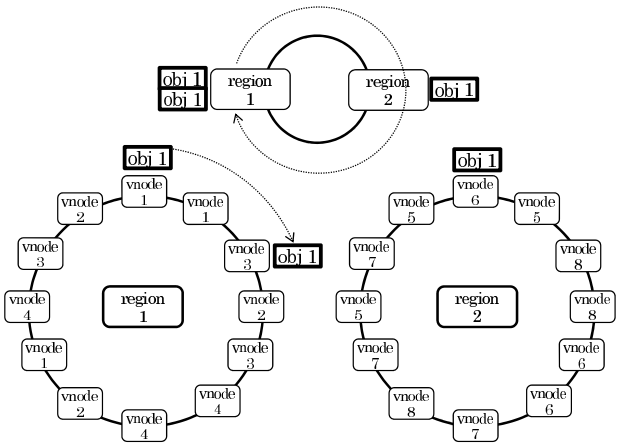

図 4 拠点を考慮したオブジェクトの配置

4.2 複数拠点へのオブジェクト配置によるデータ損失回避

従来の Sheepdog でも zone ID によるオブジェクトの集中を回避する機能を用いて，例えば 2 拠点それぞれユニークな zone ID を 2 つずつとし，オブジェクトの複製数を 3 とすることで，必ず複数の拠点にオブジェクトを配置することが可能である．しかし，それでは本来の目的である同一ラックや同一電源の物理ノードへの集中を回避したい場合と排他になってしまうし，拠点単位での障害が起こった場合はユニークな zone ID の数が設定したオブジェクトの複製数を下回ることになるため，残った拠点において冗長度の回復ができなくなる．

そのため，zone ID の他に拠点を意味する region ID を物理ノードに付与できるようにした．そして，コンシステントハッシュ法におけるリングは拠点毎に一つずつ作成し，それをネストする形で新たに仮想拠点を並べたリングを作成し，二重のリングとして管理するように変更（図 4）した．オブジェクトの配置先を決定する際はまず仮想拠点リングから配置すべき拠点と各拠点に配置するオブジェクトの個数を決定し，その後拠点毎のリングを用いて従来と同様の方式で配置先の物理ノードを決定する．これによって，一つのオブジェクトは必ず複数の拠点に分散して配置されるため，拠点単位の障害が発生によってある拠点のノードが保持するオブジェクトが全て失われても，必ず他の拠点に複製が存在することになり，データの損失を回避することができる．また，先に region に配置するオブジェクトの数を決定してから，region 内での配置先ノードを決定しているため，ノードの追加・削除によって region 内の配置先ノードが変化しても，他の region 内の配置先に影響せず，拠点をまたぐオブジェクトの移動を抑えることが可能である．

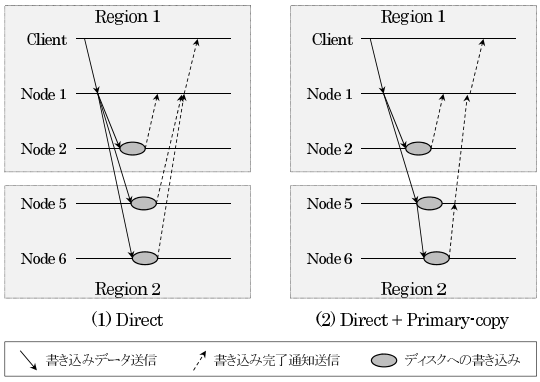

図 5 従来/広域化版 Sheepdog の書き込み処理

4.3 I/O による拠点間ネットワークトラフィック削減

従来の Sheepdog では，仮想ディスクからデータを読み込む際にアクセスされたノードは，読み込み対象となるオブジェクトを持つノードリストに自ノードが含まれない場合，ランダムに一つのノードを選んで読み込み要求を送る．しかし，アクセスされたノードと同一の拠点内のノードに対象となるオブジェクトが存在するにも関わらず，遠隔拠点のノードに読み込み要求を送ってしまう可能性があるため非効率である．広域化版 Sheepdog では，ノードリスト内にアクセスされたノードと同一拠点のノードがある場合，そのノードを優先して読み込み要求を送るように変

更した．オブジェクトの複製数が 3 で拠点数が 2 の場合，2 つの拠点には必ず同じオブジェクトが 1 つないし 2 つ配置されているため，読み込み時には拠点間ネットワークトラフィックは発生しない．

従来の Sheepdog では，仮想ディスクへのデータの書き込み時に，図 5(1) のように遠隔拠点の 2 つ以上のノードにデータの書き込み要求を送ってしまう可能性があり，拠点間ネットワークを全く同じデータが複数通過することになるため非効率である．広域化版 Sheepdog では，書き込み対象となるオブジェクトを持つノードリストに，一つの遠隔拠点に所属するノードが 2 つ以上含まれている場合，図 5(2) のように，それらのノードのうちの一つを代表として書き込み要求を送る．代表となったノードは自身と同じ拠点内のノードに書き込み要求を送り，全ての書き込み完了応答を待ってから応答を返す primary-copy 方式 [14] で複製を行うことにより，拠点間ネットワークのトラフィック量を削減している．

4.4 リカバリによる拠点間ネットワークトラフィック削減

オブジェクトのリカバリ処理においても，仮想ディスクからのデータ読み込みを行う場合と同様に，最新の epoch で保存されたオブジェクトかつアクセスされたノードと同一拠点のノードから優先的にリカバリを行うことで，拠点間ネットワークのトラフィック量を削減している．

5. 評価実験

Sheepdog によって 2 拠点にまたがる一つの分散ブロックストレージを構築した場合に，拠点間の広域ネットワークによる遅延が仮想ディスクの性能にどのような影響を与えるのか，また，拠点間ネットワークのトラフィック削減が有効であるか，そして，広域化版 Sheepdog の拠点を考慮したデータ配置が拠点単位の大規模障害に有効であるかを確認するため評価実験を行った．

表 1 実験環境

物理 PC サーバ CPU	Xeon E5-2640(6core) 2.5GHz x2
物理 PC サーバ メモリ	32GB
物理 PC サーバ NW	10GbE
物理 PC サーバ OS	Linux 2.6.32 (64bit)
物理 PC サーバ ストレージ	RAID controller (1GB cache) nearline SAS 7200rpm x16
仮想 PC サーバ HV	VMware ESXi 5.1
仮想 PC サーバ CPU 数	2
仮想 PC サーバ メモリ	4GB
仮想 PC サーバ OS	Linux 2.6.32 (64bit)
NW エミュレータ	10GbE

実験に利用した機器のスペックは表 1 の通りである．また，実験環境の全体構成を図 6 に示す．1 台の 10GbE スイッチを VLAN 機能によって 2 つのネットワークに分離することで擬似的な拠点として，VLAN 間の通信はネットワークエミュレータを経由する構成とし，各 VLAN にはそれぞれ 3 台の物理 PC サーバを接続した．各 VLAN の物理 PC サーバのうち，2 台は Sheepdog を動作させるサーバとし，1 台はクライアントとして動作させるサーバとして VMware ESXi を導入した．ネットワークエミュレータは通過する通信に対し，IP アドレス・MAC アドレス・プロトコル等の条件別に，0.04〜380ms の範囲の遅延時間を追加することが可能である．片方の VLAN には別途 1GbE の NIC を備えた PC サーバを接続して，Zookeeper ノードのみを動作させる第 3 拠点サーバとし，ネットワークエミュレータ内の設定によってそのサーバの通信は VLAN をまたいでも遅延時間が追加されないようにした上で，netem を利用して別途 100ms の遅延を追加するように設定した．各 Sheepdog 用のサーバでは，ファイルシステムとして xfs でフォーマットしたローカルディスク 2 本を利用して 2 プロセスの Sheepdog デーモンを起動し，Sheepdog の仮想ディスクにアクセスするための iSCSI ターゲットも動作させて，全体で合計 8 ノードの Sheepdog クラスタを構築した．また，それぞれのサーバにおいて Zookeeper を動作させ，第 3 拠点の Zookeeper と合わせて 5 ノードの Zookeeper クラスタを構築した．クライアントとして動作させる PC サーバではローカルディスク内のディスクイメージをシステムディスクとして仮想マシンを起動し，iSCSI イニシエータを利用することで Sheepdog から提供された仮想ディスクを追加ディスクとしてアクセス可能にした．

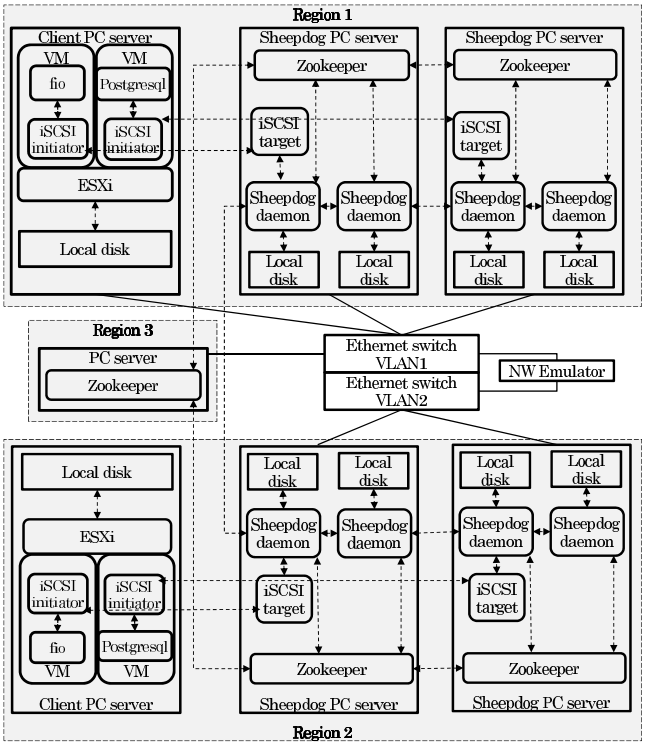

図 6 実験環境の構成

表 2　従来/広域化版 Sheepdog が提供する仮想ディスクの拠点間遅延時間別 I/O 性能 (MB/s)

	0ms(0km)	0ms(0km)		5ms(100km)		10ms(200km)		25ms(500km)		50ms(1000km)	
io pattern	local disk	normal	geo	normal	geo	normal	geo	nomal	geo	nomal	geo
512B read	1.89	0.84	0.94	0.20	0.92	0.11	0.95	0.05	0.91	0.03	0.95
4KB read	13.05	6.21	7.07	1.57	6.97	0.91	7.02	0.38	6.83	0.20	7.21
32KB read	57.76	32.08	34.03	10.72	34.33	6.62	33.53	1.67	34.33	0.90	35.37
128KB read	113.08	64.07	68.52	32.94	74.86	7.39	75.09	2.24	73.35	1.23	76.46
512KB read	133.12	89.50	90.46	23.95	96.84	9.45	99.29	3.30	99.00	1.97	99.97
1MB read	150.68	111.68	113.10	30.56	120.35	12.93	118.61	5.43	120.45	2.81	119.44
512B write	0.28	0.35	0.32	0.08	0.07	0.04	0.04	0.02	0.02	0.01	0.01
4KB write	2.43	2.69	2.41	0.63	0.57	0.35	0.32	0.15	0.14	0.08	0.07
32KB write	15.83	15.18	13.95	4.54	4.17	2.70	2.45	1.16	0.60	0.33	0.30
128KB write	38.28	37.42	35.27	15.69	14.09	9.72	3.47	0.98	0.86	0.40	0.38
512KB write	68.25	55.31	51.87	32.97	14.70	15.67	3.30	1.40	1.14	0.72	0.66
1MB write	82.76	65.94	60.77	28.41	11.86	5.74	3.96	1.80	1.51	0.93	0.88

5.1　ネットワーク遅延による仮想ディスクの性能への影響

従来の Sheepdog は複製の書き込みに同期方式を採用しており，広域化版 Sheepdog はそれをそのまま遠隔複製の方式としている．そのため，拠点間ネットワークの距離による遅延時間が，2 拠点にまたがる Sheepdog の仮想ディスクの I/O 性能に直接影響し，性能を低下させると考えられる．仮想ディスクを利用する仮想マシンやアプリケーションがどの程度の I/O 性能を得られるのかを調べるため，クライアント用サーバ上の仮想マシン 1 台において，ネットワークエミュレータによる追加遅延時間を変えながら，従来の Sheepdog と広域化版 Sheepdog が提供する仮想ディスクのベンチマークを行った．拠点間を光ファイバで接続する場合，100km あたりの往復遅延時間はおおむね 1ms であるが，一般的に途中の伝送機器等の影響も含めて 100km で最大約 5ms と言われており，実験では最悪値を得るため後者を採用している．

実験条件を一定にするため，ベンチマークの対象となる Sheepdog の仮想ディスクは容量を 10GB とし，コピーオンライトではなく，予め全データ領域を物理的に確保するプリアロケートモードで作成した．ベンチマーククライアントとなる仮想マシンでは，ソフトウェアの iSCSI イニシエータを利用して Sheepdog 用サーバの iSCSI ターゲットに接続し，仮想マシンの追加ブロックデバイスとした上で，fio を利用してファイルシステムを作らず直接ベンチマークを実行した．fio の設定により，毎回 fsync を行うランダムアクセスを，様々な I/O バッファサイズのパターンで行った場合のスループットを測定した．

表 2 にその結果を示す．表内の数値はそれぞれ同じパターンを 3 回実行した場合の平均値である．左端の列は iSCSI ターゲットのバッキングストアを，ローカルディスク内の 10GB のファイルにした場合であり，その他の列はネットワークエミュレータに設定した追加遅延時間別に従来の Sheepdog（normal）と広域化版 Sheepdog（geo）の仮想ディスクにした場合のスループットを様々なバッファサイズで測定した結果である．

まず，Sheepdog の仮想ディスクはローカルディスクと比較して，read 性能で劣り，write 性能はバッファサイズが大きい場合に劣る性能となることが分かる．Sheepdog の仮想ディスクの read 性能がローカルディスクより劣るのは，read がローカルディスクだけでなく，別の PC サーバからネットワーク越しに行う場合があることが原因と考えられる．また，I/O バッファサイズが大きい場合に write 性能が劣る原因として，fio のバッファサイズが大きくとも，OS 等によって複数の I/O に分割され，PC サーバ間の通信回数が 1 往復よりも多くなっていることが考えられる．

スループットの他に平均レイテンシも取得した．バッファサイズが 4KB までは平均レイテンシの値が，追加遅延時間に元の PC サーバ間の遅延時間を加えた値とほぼ同じとなった．バッファサイズが大きい場合には平均レイテンシが拠点間遅延時間の 2 倍以上の値となっており，ディスクのシークタイムを考慮しても 2 往復以上を要していると考えられる．また，実測は実施していないが，物理 PC サーバの RAID コントローラに搭載されているキャッシュによってシークタイムが低減されていると考えらる．Sheepdog のような分散ブロックストレージでは PC サーバ間のネットワーク遅延を含む合計遅延時間を少しでも削減するため，物理ストレージにバッテリ保護機能付きのキャッシュがあることが望ましい．

広域化版 Sheepdog の仮想ディスクの読み込み性能は，従来の Sheepdog と比較して拠点間の遅延時間が大きくなっても低下しないことが分かる．これは，アクセスされたノードと同一拠点内のノードからの読み込みを優先するように実装した効果であると考えられる．別途，従来の Sheepdog と広域化版の Sheepdog それぞれが提供する仮想ディスクから 5GB のデータを読み込んだ場合に，拠点間ネットワークを通過したデータの量をネットワークエミュレータから取得したところ，従来の Sheepdog は 1.7GB，広域化版 Sheepdog では 0GB となった．従来の Sheepdog は複製が一様に 8 ノードに分散しており，拠点外のノードが読み込み対象として選ばれる確率が 14 分の 5 のため，拠点間ネットワークの通過量の理論値は約 1.8GB であり，測定値は妥当である．広域化版 Sheepdog では拠点外から

の読み込みを抑制できたため測定値 0GB が得られたと考えられる．

広域化版 Sheepdog の仮想ディスクの書き込み性能は，従来の Sheepdog と比較して劣る結果となっている．これは，広域化版の仮想ディスクへの書き込みは必ず拠点をまたぐ通信が発生するのに対し，従来の Sheepdog ではアクセスされたノードと同じ拠点内にオブジェクトの複製 3 つを全て配置する場合があることが原因であると考えられる．別途，仮想ディスクに 5GB のデータを書き込んだ場合の，拠点間ネットワークを通過したデータの量を取得したところ，従来の Sheepdog は 7.6GB，広域化版の Sheepdog では 5GB となった．従来の Sheepdog は複製数が 3 のため，合計で 15GB を 8 ノードに分散して書き込むが，その半分は拠点外の 4 ノードに配置されることになるため，通過量の理論値は 7.5GB となり，測定値 7.6GB は妥当である．また，14 分の 1 の確率で同一拠点内に全ての複製が配置されることになる．広域化版 Sheepdog では 2 拠点に必ず 1 つ以上の複製を配置し，かつ 2 つの複製を拠点外に配置する場合は primary-copy 方式で書き込むため，書き込んだデータ量と拠点間ネットワークの通過量がほぼ一致する結果は妥当である．

表 3 は 2 台のクライアント用 PC サーバにおいて，それぞれ 2,4,8 台の仮想 PC サーバを動作させ，各仮想 PC サーバ上の iSCSI イニシエータから，同一拠点内の Sheepdog 用 PC サーバそれぞれの iSCSI ターゲットに均一に分散されるように接続し，バッファサイズ 1MB のシーケンシャルアクセスを行う fio ベンチマークを，同時並列に実行した場合のスループットの平均値を示している．

読み込みアクセスでは並列度を大きくすると，平均スループット値が低下する傾向にあり，ネットワークやクライアント側の CPU 負荷がボトルネックとなることが原因として考えられる．例外として，従来の Sheepdog かつ拠点間遅延時間が 50ms の場合は逆に並列度を高めるほどスループットが大きくなっている．その原因として仮想マシン毎のスループット値の分散は大きいため 3 回試行の平均では不足しているか，同一拠点内のノードからオブジェクトを読み込む割合が高くなっていることが考えられるが，詳細は分析中である．

書き込みアクセスでは，並列度を大きくしても平均スループット値に明確な影響が無いことが分かる．これは個々のスループットが小さく，拠点間のネットワークの遅延時間が支配的なボトルネックになっているためであると考えられる．

実験環境の拠点間ネットワークの帯域は 10Gbps であるため，合計スループットは約 1280MB/s が上限となり，仮に拠点間ネットワークの帯域が 1Gbps の場合は合計 128MB/s が上限となる．広域化版 Sheepdog の場合は拠点間距離 200km かつ 8 並列までであれば，拠点間の 10Gbps の回線はオーバースペックであり，1Gbps であっても構わない計算になる．拠点間をつなぐ広域ネットワークの敷設は非常に高コストであるため，広域版 Sheepdog を用いて 2 拠点をまたぐ分散ブロックストレージを構築する場合は，動作するアプリケーションの許容遅延時間や書き込みのスループット・並列度の推測値を考慮して，最適なネットワークを設計すべきである．広域版 Sheepdog では拠点間のネットワークを通過するデータ量を削減する設計となっているため，必要な拠点間ネットワークの帯域幅を削減可能であると考えられる．

仮想ディスクを ext4 でフォーマットし，それをデータ格納先とした Postgresql[9] を動作させ，pgbench を利用してベンチマークを実施した結果が表 4 である．pgbench は TPC-B[10] を模したパターンでデータベースにアクセスするベンチマークであり，1 回のトランザクションに複数回の read/write が含まれる．拠点間遅延時間が 0ms の場合と比較して，10ms〜25ms の間で毎秒のトランザクション数が 10 分の 1 以下になっていることが分かる．同期方式による遠隔複製は，拠点間の距離による遅延時間の影響で，特に書き込み時の I/O 性能が低下するため，アプリケーションのスループット・レスポンスの要件によっては拠点間の遅延時間の上限を考慮する必要があると言える．

複数の仮想マシン上でデータベースを同時並列に動作させた場合は，拠点間遅延時間による性能低下は見られたが，並列度増加による個々の処理性能の低下は見られなかった．データベースのトランザクション処理は I/O のバッファサイズが小さく，帯域が大きくならないためであると考えられる．

総合すると，2 拠点にまたがる広域化版 Sheepdog から提供された仮想ディスクは，読み込み性能が良く，書き込み性能は拠点間の遅延時間が支配的なボトルネックとなるため，アプリケーションの遅延やスループットの要件によって拠点間の距離を決定する必要があり，想定するスループットや並列度を考慮して拠点間ネットワークを設計する必要がある．

表 3 従来/広域化版 Sheepdog が提供する仮想ディスクの並列アクセス時の拠点間遅延時間別平均 I/O 性能 (MB/s)

	0ms(0km)		10ms(200km)		50ms(1000km)	
io pattern	normal	geo	normal	geo	normal	geo
4VM read	192.2	217.6	184.9	216.9	37.7	227.3
8VM read	156.0	148.9	156.5	150.6	59.9	147.0
16VM read	105.3	106.5	106.6	107.8	77.8	107.5
4VM write	103.9	93.8	16.8	13.8	3.1	1.9
8VM write	110.1	107.2	17.9	15.6	2.4	2.0
16VM write	69.4	71.7	20.3	17.9	2.4	2.5

5.2 拠点を考慮したリカバリ処理の性能

広域化版 Sheepdog では，拠点間ネットワークのトラフィック量を低減するため，ノードの追加・離脱時に拠点

表 4 従来/広域化版 Sheepdog が提供する仮想ディスク上のデータベースの拠点間遅延時間別処理性能 (tps)

0ms(0km)		5ms(100km)		10ms(200km)		25ms(500km)		50ms(1000km)	
normal	geo	normal	geo	normal	geo	nomal	geo	nomal	geo
308	289	118	114	71	69	23	23	14	14

表 5 従来/広域化版 Sheepdog のリカバリの性能

	time (sec)		traffic (GB)	
recovery pattern	normal	geo	normal	geo
2 node leave	133	389	21.2	35.5
2 node add	348	493	50.0	0.0

をまたぐオブジェクトの移動が大きくならないように設計されている．

表 5 は拠点間の遅延時間を 10ms（200km）とし，プリアロケートモードで作成した 100GB の仮想ディスクを格納した 8 ノードの Sheepdog クラスタから，片方の拠点の 2 ノードを停止した場合と，オブジェクトを削除した上でその 2 ノードを再びクラスタに追加した場合に，それぞれのリカバリにかかった時間と，拠点間ネットワークを通過したデータの量を示している．

仮想ディスクの複製数は 3 のため，100GB の仮想ディスクがある場合は 8 ノードのクラスタ全体で合計 300GB のオブジェクトが保存されており，1 ノードあたりの消費容量は平均 37.5GB である．広域化版 Sheepdog では両拠点に必ず同じオブジェクトが保存されているため，片方の拠点で 2 ノードの離脱があった場合は，離脱があった拠点において失われたオブジェクトをもう片方の拠点から読み込むデータの量が，拠点間ネットワークのトラフィック量となる．この場合，片方の拠点から 75GB のオブジェクトが失われ，半分はその拠点内では離脱した 2 ノードのみが保持していたオブジェクトとなるため，理論的には 37.5GB 分は拠点をまたいでリカバリを行う必要があり，実測値 35.5GB は妥当であると考えられる．2 ノード追加時は，追加された拠点内でのみデータの移動が発生し，拠点をまたいだリカバリは行われないため，拠点間ネットワークのトラフィックは発生しない．

広域化版の Sheepdog は従来の Sheepdog と比較してノード追加・離脱時のリカバリ所要時間が増加している．これは広域化版の実装にあたって，オブジェクトの複製数以上のノードが同時に離脱する拠点単位の障害においても正しくリカバリを行うため，リカバリを行うオブジェクトが最新の epoch で保存されたものかどうかのチェックや，拠点をまたいだオブジェクトのハッシュ値の取得によるオーバーヘッドが原因であると考えられ，リカバリ速度の改善は今後の課題となっている．

5.3 拠点障害時のフェイルオーバー

データベースのデータ格納先を広域化版 Sheepdog の提供する仮想ディスクとし，拠点単位の障害が発生した場合に，残った拠点で障害直前のトランザクションを維持したデータベースのフェイルオーバーをいわゆる Active-Standby 構成によって実現可能かどうか，障害からデータベース再開までの所要時間を実験によって確認した．

実験は拠点間遅延時間を 10ms（200km）とし，以下の手順で実施した．

(1) 拠点 1 において，クライアント用 PC サーバの仮想マシン上で，Sheepdog から提供された仮想ディスクをデータ格納先とした Postgresql を実行

(2) Postgresql のテスト用テーブルに対してランダムに生成したレコードを INSERT するクエリを発行し，成功した場合はシステムディスク内に INSERT 済レコードとして記録することを連続的に行うスクリプトを実行

(3) そのスクリプトを開始して 60 秒後に，拠点 1 の全ての Sheepdog 用 PC サーバにおいてキャッシュフラッシュを伴わないリブート（reboot -nfh）を実行（拠点単位の障害を再現）

(4) 拠点 2 で，予めフェイルオーバー用の仮想マシンを起動しておいて，Zookeeper に対して Sheepdog のノードリストをポーリングするスクリプト（フェイルオーバースクリプト）を実行させておき，拠点 1 に所属する全ての Sheepdog ノードの離脱を拠点単位の障害として検知

(5) 拠点単位の障害を検知した場合，拠点 1 で Postgresql がデータ格納先としていた仮想ディスクに接続し，Postgresql を実行

(6) 拠点 2 で実行されている Postgresql のテスト用テーブルに新規にレコードを INSERT できることを確認

(7) 拠点 1 において記録しておいた INSERT 済のレコードと，拠点 2 の Postgresql から SELECT によって取得したレコードを照合

実験の結果，従来の Sheepdog では拠点単位の障害によって，オブジェクトの複製数 3 以上となる 4 ノードが離脱したために，障害前に Postgresql がデータ格納先としていた仮想ディスクを利用しようとするとエラーとなり，アクセスすることができなくなった．広域化版 Sheepdog では障害前に Postgresql がデータ格納先として利用した仮想ディスクを生き残った拠点からアクセス可能で，テスト用テーブルに継続して INSERT が可能，かつ障害拠点において記録しておいたテスト用テーブルに INSERT 済レコードとフェイルオーバー後に SELECT で得られたレコードが一致，もしくはフェイルオーバー後の方が 1 レコード多い状態となった．Postgresql はクエリ毎に fdatasync によってディスクへの永続化を行うようになっており，そのディスクは同期方式で遠隔複製された仮想ディスクであるため，

障害直前までの書き込みが反映（RPO がゼロ）されていたと考えられる．記録よりも 1 レコード多い状態となった原因は，ディスクへの永続化が完了したものの，クライアントに書き込み完了の返答をする間に障害が発生したためであると考えられる．

表 6 フェイルオーバーのタイムテーブル

経過時間 (sec)	差分時間 (sec)	状態
0	-	障害発生
5.5	5.5	フェイルオーバースクリプト開始
9.2	3.7	iSCSI ターゲットに接続完了
10.9	1.7	ext4 マウント完了
13.3	2.4	Postgresql への INSERT 再開

擬似的に障害を発生させてから各状態となるまでの平均経過時間は表 6 の通りである．拠点全体の PC サーバの障害ではなく，拠点間ネットワークの切断を模擬し，Zookeeper の leader ノードを配置した優先拠点でフェイルオーバーを行う場合も同様に測定したが，平均値は表 6 と同じ値であった．

障害からフェイルオーバースクリプトが起動するまでの時間には，Sheepdog ノードの離脱を検知するための所要時間として，Sheepdog デーモン起動時に設定した Zookeeper に対するタイムアウト値を 1 秒と，Zookeeper の Heartbeat 間隔 1 秒，さらにフェイルオーバースクリプトの Zookeeper からのノードリストポーリング間隔である約 1 秒が含まれていると考えられる．Sheepdog デーモンの Zookeeper に対するタイムアウトや Zookeeper の Heartbeat 間隔を短く設定することによって，この区間の所要時間を短縮できると考えられるが，予期せぬ遅延によって iSCSI ターゲットに接続するための所要時間約 3 秒は必要経費である．追加ディスクを ext4 マウントするためにかかる時間には，ファイルシステムのリカバリ時間が含まれていると考えられる．ファイルシステムのリカバリではブロックストレージの特定領域への読み込み・書き込みが発生する．Postgresql の起動から INSERT 再開までの時間はデータベースの起動処理とリカバリ処理を行っている時間であると考えられる．実験ではテーブルが空の状態から INSERT を開始したが，既に大量のレコードが入った状態のデータベースをリカバリする場合は，起動から最初の INSERT を受け付けるまでの時間が長くなる可能性がある．

広域化版 Sheepdog を利用して拠点をまたいだデータベースのフェイルオーバーを行う場合，RPO はゼロ，RTO はファイルシステムの再開まで 10 秒強，データベースの再開まで 14 秒以上であると言える．また，RPO がゼロとなる条件として，アプリケーションがキャッシュを利用せず，I/O の度に fdatasync 等によって確実にディスクに書き込みが行われている必要がある．

6. 関連技術

Sheepdog は本論文において実装のベースとした分散ブロックストレージであり，オープンソースの分散ブロックストレージである Ceph と異なる点として，メタデータサーバ・データサーバのような構成するノードの役割分担が無いことが挙げられる．

EMC VPLEX Metro は拠点間距離の最大値を 200km として同期方式の遠隔複製を実現している．巨大なバッテリ保護キャッシュを持つ FC スイッチのような実装であり，複数の SAN ストレージ製品を束ねて管理することが可能で，2 拠点間で遠隔複製を行う仮想ディスクを設けることができる．特徴として，2 拠点間のキャッシュの一貫性を保つための分散ロック機能を持っており，本論文における広域化版 Sheepdog は一つの仮想ディスクに対してどちらかの拠点からのみアクセス可能であるのに対して，VPLEX Metro が提供する仮想ディスクは 2 拠点で同時並列にアクセスすることが可能である．そのため，より安全に拠点をまたぐ仮想マシンのライブマイグレーションを行うことが可能であり，また，OCFS2 や GFS を例とする分散ファイルシステムや Active-Active 動作が可能なデータベースを拠点をまたいで利用可能であると考えられる．

DRBD はブロックストレージの同期方式による遠隔複製を実現可能なオープンソースソフトウェアである．VPLEX Metro と同様に 2 拠点で同時並列のアクセスが可能なデュアルプライマリ機能を持っており，さらに拠点間ネットワークの帯域幅がボトルネックとなる場合にデータのバッファリングや圧縮を行うことができるオプションも存在する．DRBD や VPLEX Metro は構成する物理リソースの範囲で行った静的な設定によって容量・性能が決定されるため，動的なリソース追加・縮退が難しい．本論文では，動的に容量・性能を拡張・縮退することが容易な分散ブロックストレージである Sheepdog をベースとして遠隔複製機能を追加することにより，その問題を解決している．

7. まとめ

本論文では，柔軟なスケール性を持ちつつ，同期方式の遠隔複製が可能な分散ブロックストレージである広域化版 Sheepdog について述べた．広域化版 Sheepdog の仮想ディスクを利用するとき，拠点間ネットワークによる遅延時間が遠隔複製される仮想ディスクの性能に与える影響を確認し，読み込み性能に影響しないこと，書き込み性能が遅延時間によって著しく減衰すること，スループットの減衰とアクセスの並列度を考慮した拠点間ネットワークの帯域幅設計が必要なことを示した．また，従来の Sheepdog では対応不可能だった拠点単位の大規模障害や拠点間ネットワークの切断が発生した場合でも，適切な拠点においてデータベースのフェイルオーバーが可能であることを実験

によって示した.

今後の課題として，5.2 章で挙げたリカバリ速度改善の他に，複数拠点間において 1 つの仮想ディスクに対する同時アクセスを可能とし，分散ロックマネージャ機能を含むアプリケーションであれば Active-Active 運用を可能とすることと，1 つの Sheepdog クラスタ内で拠点をまたぐ遠隔同期の対象とする仮想ディスクと特定の 1 拠点にのみ複製を配置する仮想ディスクを混在可能にすることを検討している.

参考文献

[1] 森田和孝, 藤田智成, and 盛合敏. ”Sheepdog: 仮想マシンのための対称型クラスタストレージ (コンピューティングシステム Vol.5 No.2).” 情報処理学会論文誌 論文誌トランザクション (2012):23-32.
[2] Weil, Sage A., et al. ”Ceph: A scalable, high-performance distributed file system.” Proceedings of the 7th symposium on Operating systems design and implementation. USENIX Association, 2006.
[3] DRBD, http://www.drbd.org
[4] EMC. EMC VPLEX Metro, http://japan.emc.com/storage/vplex/vplex.htm
[5] VMware. Virtual SAN, http://www.vmware.com/jp/products/virtual-san
[6] tgt project. Linux SCSI target framework(tgt), http://stgt.sourceforge.net
[7] QEMU community. QEMU, http://wiki.qemu.org
[8] fio project. fio, http://freecode.com/projects/fio
[9] Oracle. PostgreSQL, http://www.postgresql.org
[10] TPC. TPC-B, http://www.tpc.org/tpcb
[11] Hunt, Patrick, et al. ”ZooKeeper: wait-free coordination for internet-scale systems.” Proceedings of the 2010 USENIX conference on USENIX annual technical conference. Vol. 8. 2010.
[12] Dake, Steven C., Christine Caulfield, and Andrew Beekhof. ”The corosync cluster engine.” Linux Symposium. Vol. 85. 2008.
[13] Karger, David, et al. ”Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web.” Proceedings of the twenty-ninth annual ACM symposium on Theory of computing. ACM, 1997.
[14] Alsberg, Peter A., and John D. Day. ”A principle for resilient sharing of distributed resources.” Proceedings of the 2nd international conference on Software engineering. IEEE Computer Society Press, 1976.
[15] EMC Education Services. ”ストレージの原則と技術” (p.275-295), 株式会社インプレスジャパン, 2013.