Vol.2013-IOT-20 No.20
2013/3/14

大学間連携による頑強な広域分散データ基盤アーキテクチャの提案

中川 郁夫[1,2]　橋本 好史[1]　楠田 友彦[1]　北口 善明[3]　近堂 徹[4]　柏崎 礼生[2]　市川 昊平[5]
下條 真司[2]

概要：本稿では, 複数大学のコンピュータリソースを相互に接続し, 広域分散環境において頑強なデータ基盤を構築するためのアーキテクチャの提案を行う. 本アーキテクチャは, 多数のコンピュータリソースをもちいて大規模なストレージ空間を実現するクラスタストレージ技術を応用し, 広域分散環境で信頼性の高いデータ基盤を実現する. 本アーキテクチャによって実現されるデータ基盤は, 地理的に離れた 3 カ所以上の「マルチサイト」から同時にアクセスが可能であり, かつ一カ所での更新がリアルタイムに他サイトに反映されることを特徴とする. 同データ基盤は単一の巨大なファイルシステムとして POSIX 準拠の標準的なアクセス手段を提供し, かつ, ランダムアクセスによるファイル入出力を可能とする. 将来的には, 地理的に離れた拠点間で仮想 OS のライブマイグレーションができることを目指す.

キーワード：広域分散ストレージ, クラスタストレージ, ライブマイグレーション, Disaster Recovery

1. はじめに

近年, 大学・研究機関はもちろん, 民間企業においても, 大規模データの保存や処理に対するニーズは急速に強まっている. 計算機 (コンピュータ) で扱う電子データは情報機器増, 種類増, 機会増, 詳細化 (高解像度化, 精細化, 精密化) などを背景に急増し, 人類が扱うデータの総量は, 2015 年から 2025 年の 10 年間で 130EB(エクサバイト) から 7916EB ≒ 8ZB(ゼッタバイト) に増加すると予測されている [1]. また, そのほとんどのデータが再利用, 統計・解析の対象となることから, 大規模データの保存・処理技術がさらに重要になることは明らかである.

大学, 研究機関の研究活動においても大規模データを解析するための技術研究や環境整備は重要である. すでに, 米国では大規模データ処理に関わる研究に 2 億ドル規模の予算を投じている. 国内でも, いくつかの大学で大規模データ処理基盤の構築に着手しているが, 現状の, 個々の大学の取り組みでは次のような課題があることも指摘されている.

- 個々の大学の予算では, 容量や性能面で十分な規模を実現できない.
- 耐障害性の向上や災害対策のため, 地理的に分散してデータを冗長化する仕組みが別途必要である.

本稿では, 上記課題を解決する仕組みを実現するため, 複数の大学が有する多数の計算機 (コンピュータ) をクラスタ化することにより, 広域分散環境に大規模で信頼性の高いデータ基盤を実現するためのアーキテクチャの提案を行う. 本稿で提案する広域分散型のデータ基盤アーキテクチャは, 以下を特徴とする.

- 複数大学が有する多数の計算機 (コンピュータ) をクラスタ化することで「規模」を実現し, 単一組織ではなし得なかった, 巨大な容量や性能を持つ仮想的なデータ空間を実現する.
- 地理的に離れた場所に自動的にデータを分散保存することで, 障害や災害時にも強い「頑強」なデータ空間を実現する.

なお, 本稿では, 広域分散型のデータ基盤アーキテクチャの中でも, 特に, データの書き込み性能に着目し, 1 拠点でデータを書き込む場合と同程度の書き込み速度を実現しつつも, 地理的にも離れた, 広域分散環境で即時に変更が反映される仕組みを新たに提案する.

本稿では, 第 2 章で先行研究について述べる. また, 第 3 章で本研究におけるデータ基盤に関する要求を実現コンセプトとしてまとめ, 第 4 章で提案するアーキテクチャについて述べる. 第 5 章で課題や今後の取り組みについて述べ, 第 6 章でまとめる.

[1] 株式会社インテック
Intec, Inc., Koto-ku, Tokyo 136–0075, Japan
[2] 大阪大学
Osaka University
[3] 金沢大学
Kanazawa University
[4] 広島大学
Hiroshima University
[5] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

2. 先行研究

クラスタ型のストレージ技術は, 近年のクラウドコンピューティングの普及とともに, 並列分散コンピューティングを応用したスケールアウト型のコンピューティング技術として急速に発展した.

スケールアウト型のアーキテクチャに関しては, Google によるクラウドプラットフォーム技術が先行する. Google は, 300 万台とも 500 万台とも言われる, 膨大な数のコンピュータを用いて現在の Google のサービスを支えるアプリケーション基盤を実現している. そこでは、大規模なファイルシステムである GFS [2] や, 独自のデータベースシステムである BigTable [3], あるいはデータ処理機構である MapReduce [4] が運用されている. これらの技術は、既に実績もあり、広域分散環境で信頼性高いサービスを実現している. 一方, これらの技術は, いずれも Google の内部技術であり, すべての Google アプリケーションが専用の API やプロトコルを必要とする. また, GFS は追記型ストレージであり, ファイルの部分的な更新などは不得手である.

Amazon S3 や同サービスと互換性を特徴とするストレージサービスも発表されている. S3 は Dynamo [5] と呼ばれる分散 KVS を用いて分散環境でデータを保存する仕組みをストレージサービスに応用したものである. S3 はオブジェクトストレージに分類される. オブジェクトストレージはクラウド型のサービスとして, Basho をはじめ多数のサービスが実用化されている. なお, オブジェクトストレージは, 入出力が単純なデータブロックの get と put (もしくは set) のみであり, アプリケーションが REST/HTTP などの専用の API を利用する必要がある.

その他, 拡張機能を有するクラウド型のストレージとして, Google Drive, Microsoft SkyDrive, Apple iCloud, DropBox, Evernote, など多数のサービスが存在するが, いずれも専用のアクセスインターフェースを必要とするため, 既存のアプリケーションからファイルシステムとして利用するには, 何らかの変更, 改修が必要になる.

本稿では, スケールアウト型の広域分散型ストレージでありながら, 標準的な POSIX [6] ファイルシステムとして利用可能なストレージアーキテクチャを提案する. 同アーキテクチャでは, 利用者は, NFS などを介して標準的なファイルシステムとしてアクセスが可能である. また, ランダムアクセスによるファイル入出力サポートにより, 既存のアプリケーションに手をいれず従来と同様に利用でき, かつ, 仮想 OS からも利用可能なストレージを実現することを目指す.

3. コンセプト

以下では, 本稿で提案する広域分散データ基盤が目指す, ストレージサービスのコンセプトについて整理する.

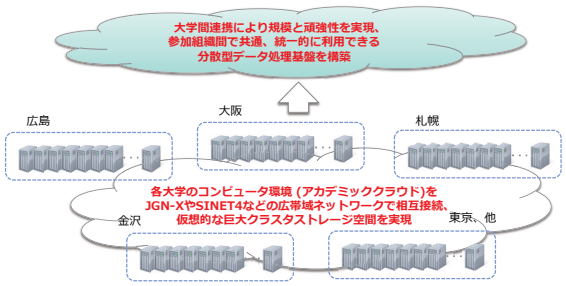

図 1 コンセプト
Fig. 1 Concept

本稿で提案する広域分散データ基盤は, 複数大学の計算機を相互に接続した環境に構築される. これらの計算機のリソースはソフトウェアにより制御され, 全体としてひとつのストレージ空間を実現する. 同ストレージ空間は, 各大学から共通のデータ基盤として利用されるため, 以下のような機能を持つことが求められる.

3.1 Multi Location ($N > 2$)

複数の地理的に分散された場所で共通のストレージ空間を構築する. 一カ所で本ストレージ空間に書き込まれたデータは, 他の拠点で複製することにより信頼性を高める. 特に, N 箇所 ($N > 2$) の地理的に分散された場所にある大学のリソースを相互に接続することで, 信頼性を向上させ, 障害や災害に強い頑強なデータ基盤を実現する.

なお, DR (Disaster Recovery) 機能を持ったストレージなどでは, ある拠点のデータを他拠点でバックアップする $N = 2$ 型が多い. 本稿では, $N > 2$ の多拠点型の分散ストレージサービスを目指す.

3.2 All Active

前述 N 箇所の拠点では, どこからでもファイルの書き込み, 読み込みが可能な All Active 型を目指す. すなわち, ある拠点で書き込まれたデータは, 同時に他の拠点からも読み込みが可能になり. データの複製も, 拠点間の回線が許容する限りリアルタイムに行う.

従来型の DR サービスでは, 定常時は主系サイトでサービスを実施し, 従系サイトでバックアップを実施する, 定期的なデータコピー＋マスタ/バックアップ型 (Active/Stand-by 型) のものが多い. 従来型ではデータの複製までの時間差が大きい事, 及び同時に利用できるのが片系のみであることなどの問題がある. 本稿では, リアルタイムに複製を行い, 複数拠点が同時に利用可能な All Active 型を目指す.

3.3 Native File System

従来型の多くのアプリケーションからの利用を想定し, 標準的な POSIX 準拠のファイルシステムを提供する.

多くのクラスタストレージ, またはクラウド型のストレー

ジサービスでは, 専用のインターフェースや専用のアプリケーションを必要とする. 本稿では, 標準的なファイルシステムをサポートすることで, NFS などの既存の仕組みで利用できるストレージ空間の実現を目指す.

3.4 Random Access

ファイルの任意の場所を参照, 更新できるランダムアクセスによるファイル入出力を行う.

オブジェクトストレージやほとんどのクラウド型のストレージサービスでは, ブロックデータのシーケンシャルアクセスのみを提供し, ランダムアクセスには対応していないため, 例えば, 仮想 OS のストレージとして利用することはできない. 本稿では, ランダムアクセスをサポートすることで, 既存のすべてのアプリケーションから利用可能, かつ, 特に仮想 OS のストレージとしても利用できるストレージ空間の実現を目指す.

4. アーキテクチャ

以下では, 広域分散データ基盤を実現するためのアーキテクチャについて述べる. 本稿で提案するアーキテクチャは株式会社インテックが開発した EXAGE / Storage [8] を応用し, 大学間のコンピュータリソースの連携により広域分散型のストレージ空間を実現するものである. 本章では, まず, 複数大学によるコンピュータリソースの連携の概念についてまとめ, ネットワーク接続アーキテクチャ, 及びストレージアーキテクチャについて述べる.

4.1 大学間連携

本研究では, 複数の大学の計算機資源を相互に接続することで, 大規模なデータ基盤を実現する. 具体的には, 各大学にストレージ用の計算機資源と, ストレージを利用するアプリケーションが動作する計算機資源を準備する. 前者は, 広域環境でのストレージ空間を実現し, すべての大学で共通の単一ファイルシステムを構成する. 後者は, 同ストレージ上のファイルを扱うアプリケーションが稼働する計算機資源である. 例えば, 大学向けにサービスを行うコンテンツ配信システムや, 仮想 OS などを動作させるホストコンピュータなどが相当する.

4.2 ネットワーク接続アーキテクチャ

前述の大学間連携による計算機資源の相互接続のために, 本研究では, すくなくとも二つの高速・広帯域ネットワークを必要とする. 具体的には, ストレージ用の計算機資源を相互接続するためのストレージネットワーク, 及び, 各大学の処理空間を相互接続するためのアプリケーションネットワークである. ストレージネットワークは, データ基盤に保存されるユーザデータの同期に利用する. また, アプリケーションネットワークは, アプリケーション間の通信

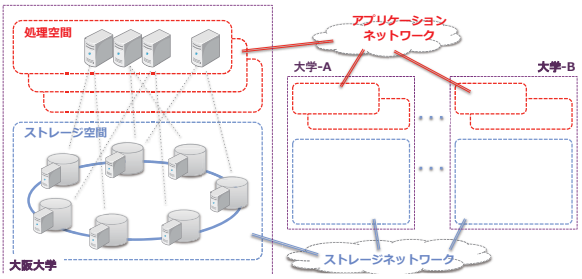

図 2 大学間連携の概念図
Fig. 2 Overview of Academic Cloud

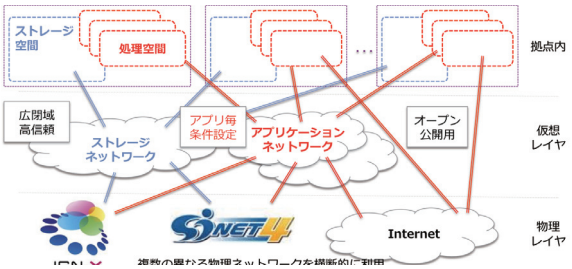

図 3 ネットワークの接続イメージ
Fig. 3 Network Implementation

や, 仮想 OS のライブマイグレーションなどで利用する.

なお, 本研究プロジェクトでは, 現在, 実証実験環境の構築を進めているが, 上記, 高速広帯域ネットワークとして JGN-X と SINET4 を活用している.

4.3 ストレージアーキテクチャ

本研究では, 既存のクラスタストレージを応用することで, 広域分散環境での頑強なデータ基盤を実現する. 以下では, クラスタストレージを実現するソフトウェアとして利用する EXAGE / Storage について述べる. 同ソフトウェアは, 並列分散コンピューティングの技術を応用し, 汎用性の高い廉価な IA サーバを多数並べることで, SPF (Single Point of Failure) が存在しない, スケーラブルで信頼性の高いストレージ空間を実現する.

EXAGE / Storage では, ストレージを利用する「クライアント」からのファイルの入出力要求を, フロントエンドである「アクセスサーバ」が受け付ける. アクセスサーバはクライアント向けに NFS などの POSIX 準拠のファイルシステムとしてデータ空間を公開する. また, ストレージ内部では「コアサーバ」と呼ばれる多数のコンピュータ上にメタデータ (管理データ) 及びユーザデータ (ファイルの中身に相当する実データ) を保存する. 図 4 に構成のイメージを示す.

4.3.1 メタデータの管理

メタデータは分散 KVS を用いてデータの保存, 更新を行う. 分散 KVS は分散 NoSQL と呼ばれる技術の一つである.

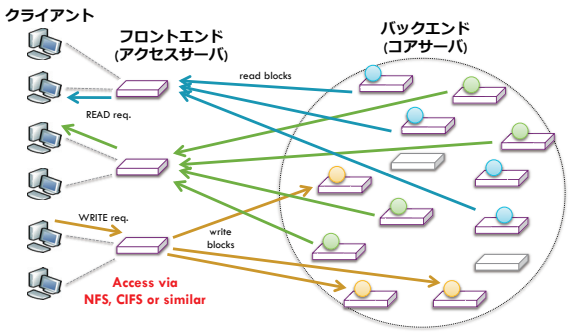

図 4　EXAGE / Storage の概要
Fig. 4　EXAGE / Storage Overview

EXAGE / Storage では, 独自の分散 KVS である EXAGE / KVS を内部的に利用しており, SPF のない, スケーラブルなメタ情報の管理を可能にする. なお, EXAGE / KVS については本稿の主題とは異なるため詳細は割愛する.

4.3.2　ユーザデータの管理

ユーザデータ, すなわちファイルの中身に相当する実データは, ファイルを分割したブロックとして, 多数のコアサーバ上に分散して保存される.

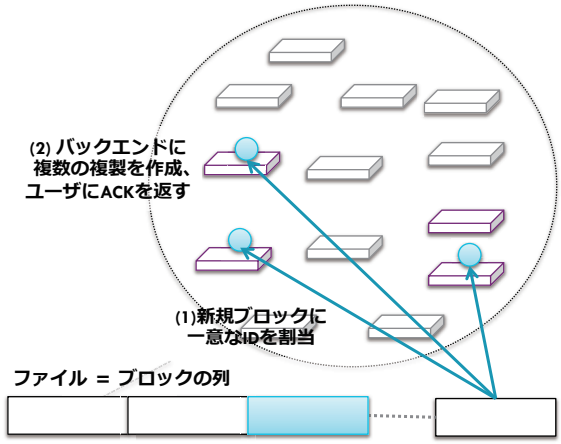

図 5　ブロックの複製ステップ
Fig. 5　Block Replication Step

図 5 は, ブロックデータの冗長化の考え方を図示したものである. EXAGE / Storage は, 各ブロックが, 常に指定の多重度 (通常は 3) 以上の, 物理的に異なるサーバ上に存在するように複製管理を行っている. ブロックを保持するサーバがダウンした場合には, 当該サーバ上にあったブロックデータを他のサーバ上に新たに作成することにより, 常に多重度が指定の値以上になるように維持する.

4.4　広域分散アーキテクチャ

本研究では, EXAGE / Storage を拡張することにより, 複数の異なる拠点にある計算機資源を用いて EXAGE / Storage を構成できるようにし, 結果的に, どの拠点からでも単一の巨大なファイルシステムにアクセスできるようにする. すなわち, 本稿で提案する広域分散型のデータ基盤は, EXAGE / Storage に対して, 異なる複数の拠点でのクラスタストレージを実現する機能である「マルチサイト」に対応することで実現する. マルチサイト対応アーキテクチャを実現するためのポイントは以下の 2 点である.

4.4.1　メタデータの複数拠点配置

複数拠点でストレージ空間を共有するためには, 複数拠点間でメタデータ情報をリアルタイムに共有する必要がある. 本研究では, EXAGE / KVS を複数拠点で共有することにより, メタデータを「マルチサイト」対応にする. なお, 本稿はユーザデータの広域分散に関するアーキテクチャに主眼を置いており, メタデータの複数拠点配置に関わるアーキテクチャの詳細については別稿で議論する.

4.4.2　ブロックデータの複数拠点配置

マルチサイト対応のためには, ユーザデータである, ブロックデータの複製を複数拠点に配置する必要がある. ブロックデータの総量は, ファイルサイズに比例して大きくなるため, ネットワーク遅延や帯域が性能に与える影響が大きくなる. 以下では, これらの影響を最小限にするため, 広域分散環境でデータを分散保存するための仕組みとして, 以下のアルゴリズムを組み込む.

(1) クライアントからのブロック作成要求に対し, ローカルのコアサーバ上にブロックとその複製を作成し, クライアントに ACK を返す. クライアントは, 自身がアクセスする拠点内にブロックデータが冗長化された状態で ACK を受け取る.
(2) コアサーバ上の複製管理機構が, 新規作成されたブロックの複製を地理的に異なる拠点に作成する. 同処理は、ローカル以外に $(M-1)$ の複製が作成されるまで繰り返される. 結果的に, システム全体では $(M+1)$ のブロックが作成される. なお M は指定された多重度を示す.
(3) 複製管理機構は, 多重度が指定された値よりも多い場合, 多重度を減らす処理を行う. 同じサイトに重複している場合を優先して削除するため, 最初にローカルで作成された 2 つのブロックデータの一方が削除される.

図 6 にマルチサイト対応のアルゴリズムを図示する. 上記アルゴリズムでは, ある拠点で書き込み要求があった場合, ローカルで複製ができた時点でクライアントに ACK を返す. 広域分散環境でのブロックデータの複製は, ACK を返した直後からバックグラウンドで行われる.

一般に, 地理的に離れた場所でデータの複製を持つ広域分散ストレージでは, ネットワーク遅延や帯域の影響から, 書き込み速度の低下を招くことが問題になる. 本アーキテクチャでは, ある程度の信頼性を確保しながらも, できるだけ早く ACK を返す”write back”型のアルゴリズムを採用することで, ローカルでクラスタストレージを構成する場合と同程度の体感性能を実現する.

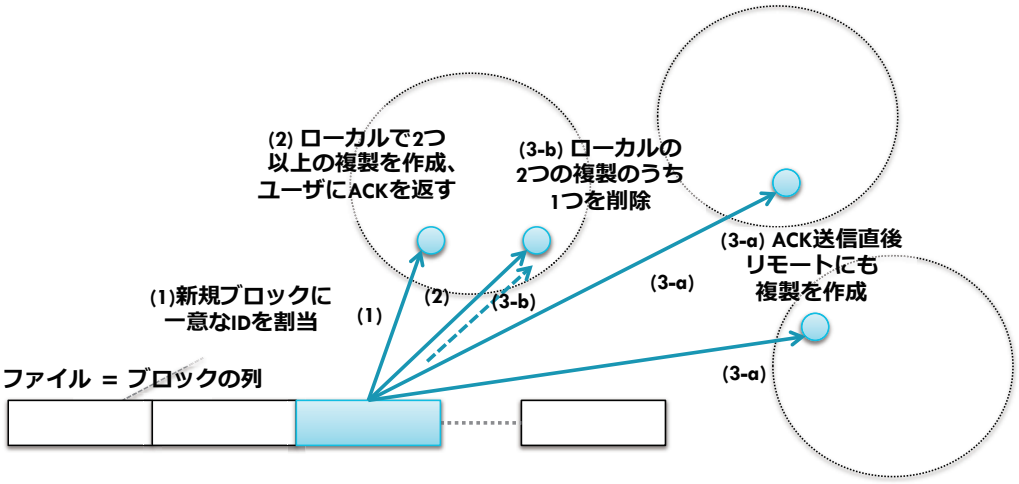

図 6　分散型の複製管理
Fig. 6　Distributed Replication Management

さらに，EXAGE / Storage では，ブロックの更新に”write-once”モデルを採用しており，変更された箇所を含むブロックは新たな ID を持つ異なるブロックとして登録される．そのため，1 拠点でデータを保存した後，ID がメタデータ上で更新された段階で変更が反映されるため，多拠点から同時にアクセスがあっても一貫性を損なうことはない．

5. 考察

以上では、本稿で提案した広域分散型のデータ基盤アーキテクチャを実用化する際の課題についてまとめる．

5.1 拠点間の遅延の影響

実際の環境では，拠点 (大学) 間には通信遅延が発生する．一般的に，日本国内では数ミリ秒〜数十ミリ秒程度，日米間で 100 ミリ秒〜200 ミリ秒程度，とされる遅延がある．これらの通信遅延は，データの更新などに大きな影響を与える可能性がある．本アーキテクチャでは，ユーザデータの書き込みはローカルで処理された後にクライアントに ACK を返すことができるため，拠点間の通信遅延はユーザからみた書き込み性能に対して直接的には影響しない．一方，メタデータは分散 KVS を用いて拠点間でデータを共有しているため，分散 KVS の仕組みによってはメタデータの更新時間に影響する可能性がある．分散 KVS の影響に関する詳細については別稿で議論する予定である．

5.2 拠点間の通信帯域の影響

拠点間の通信帯域は，ブロックデータの複製にかかる時間に大きく影響する．本アーキテクチャでは拠点間のブロックデータの複製はバックグラウンドで行われるが，あるサイトにおけるユーザからの平均書き込みスループットと同じかそれ以上の通信帯域がなければ，必要なブロックデータの複製の作成が追いつかないことは明らかである．本研究プロジェクトでは，準備する計算機資源の量に比較して，JGN-X や SINET4 の通信帯域は十分に広いため，上記要件は満たすが，より一般的な拠点間で同アーキテクチャを採用する場合には，通信帯域がユーザ向けの帯域に満たない場合の影響や運用についても検討・検証が必要である．

5.3 対障害性

本稿で提案するデータ保存の仕組みは，体感の書き込み性能を向上させる一方，対障害性に対するリスクを許容する必要がある．具体的には，本アーキテクチャでは，書き込みが成功したデータは「ローカルで冗長化」されていることは保証されているが，「グローバルで冗長化」されるには若干 (通常は数百ミリ秒から数秒) の時間差が存在する．今後の研究では，あるデータがグローバルで冗長化されたことを検知・確認する仕組みの実装なども検討が必要である．

5.4 多数サイトでのリプリケーションポリシ

サイト数 N が大きくなった場合，ブロックデータの複製の持ち方に工夫が必要である．あるサイトで書き込んだデータは，当該サイト内では原則データブロックが存在するが、障害や災害時，サイト切り替えが発生した場合には，切り替わった先のサイトにブロックデータが存在する確立は M (通常は $M = 3$) を多重度として，$(M-1)/(N-1)$ になる．すなわち，N が大きいときには，ブロックデータが存在しない可能性が高い．仕組み上，ブロックデータを持つサイトが一つでも存在すれば (当該サイトと通信が可能であれば) 当該ブロックを読み出す事ができるため読み込みエラーにはならないが，遠方サイトからデータを読み込む場合には，読み込み性能に影響することが推測される．今後は N が大きい場合のリプリケーションポリシを柔軟に定義するための仕組みも検討する．

6. おわりに

本稿では, 複数大学のコンピュータリソースを相互に接続し, 広域分散環境において頑強なデータ基盤を構築するためのアーキテクチャを提案した. 本アーキテクチャは, 多数のコンピュータリソースをもちいて大規模なストレージ空間を実現するクラスタストレージ技術を応用し, 広域分散環境で信頼性の高いデータ空間を実現する. このデータ空間は, 広域分散ストレージとして以下の要件を満たす.

(1) Multi Location ($N > 2$)
(2) All Active
(3) POSIX File System Support
(4) Random Access File I/O Support

$N > 3$ の複数拠点において同時にストレージアクセスを可能とし, 1 カ所での更新が即時に他の拠点に反映させる仕組みは, 分散型のストレージサービスとして重要である. 本稿では, 特に, ローカルでの複製の作成が完了した時点で ACK を返すことにより高速な書き込みを実現しつつ, 広域環境で一貫性を保ったまま, 即時にその更新が反映される仕組みについて説明した.

このような仕組みは, VM (Virtual Machine, 仮想化サーバ) をユーザに提供するサービスにおいて特に強いニーズがある. 例えば, 遠隔拠点間で仮想マシンのライブマイグレーション技術を用いる場合には共有ストレージの同期が必須である. 一方, 一般には, 広域分散環境でストレージの同期を行うのは容易ではないため, 長距離を隔てた地点でのライブマイグレーションに関する報告は前例がない. 本稿で提案する仕組みは, 異なる拠点間で, 同時に共通のファイルシステムにアクセスすることが可能であり, 遠隔拠点間でのライブマイグレーションの実現を可能にする.

本研究プロジェクトでは, 本稿で提案した広域分散型のデータ基盤アーキテクチャに基づいて実証実験環境の構築を進めている. 既に実証実験の環境では, JGN-X や SINET4 を用いて, 大阪大学, 金沢大学, 広島大学および国立情報学研究所を結ぶ多拠点の分散ストレージを実現している. 今後は実証実験の環境を用いて, データ基盤としての基本性能の他, 仮想 OS のライブマイグレーションの実施など, より詳細な検証, フィージビリティスタディを行うことを予定している.

謝辞 本研究にあたり, 多数の有用なコメントをいただいた distcloud プロジェクトのメンバ各位に感謝します. また, 本研究の実証実験にあたり, コンピュータリソースのご提供をいただいた各大学, JGN-X の回線をご提供いただいた独立法人情報通信研究機構, SINET4 の回線をご提供いただいた国立情報学研究所, および, クラスタストレージ技術である EXAGE/Storage をご提供いただいたインテックに感謝します.

参考文献

[1] Frank Gens: *IDC Predictions 2012 : Competing for 2020*, IDC, 2012.
[2] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung: *The Google File System*, 19th ACM Symposium on Operating Systems Principles, NY, October 2003.
[3] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber: *Bigtable: A Distributed Storage System for Structured Data*, OSDI'06: Seventh Symposium on Operating System Design and Implementation, Seattle, WA, November, 2006.
[4] Jeffrey Dean and Sanjay Ghemawat: *MapReduce: x Simplified Data Processing on Large Clusters*, OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December 2004.
[5] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogel: *Dynamo: Amazon' s Highly Available Key-value Store*, SOSP2007, 2007.
[6] *POSIX.1: IEEE Std 1003.1–2008*, http://pubs.opengroup.org/onlinepubs/9699919799/, IEEE, 2008.
[7] 平井日出美, 中川郁夫: リアルクラウドソリューション, INTEC TECHNICAL JOURNAL, No.11, pp40-45, 2011.
[8] *EXAGE — Cloud Platform Technology*, http://www.intec.co.jp/service/exage/, INTEC, INC., 2013.