Vol.2011-HPC-130 No.34
2011/7/28

クラウドにおける大規模ストレージシステムの必要性とその検討 ~Gfarm v2.4 を拡張した EB 級システム~

永岡　孝[†]　石津　晴崇[†]　大西　健司[†]
高杉　英利[†]　建部　修見[††]

近年，エクサバイト級のストレージシステムの必要性が現実的になってきている背景をふまえ，本研究報告では現状のストレージシステムでの容量拡張の限界について考察した．また，Gfarm v2.4 をベースとした，エクサバイト級のストレージシステムを実現する設計を検討し，プロトタイプシステムを作成し，その可能性を確認した．

Necessity and examination of large-scale storage system in Cloud －System of ExaBytes class that enhances Gfarm v2.4－

Takashi NAGAOKA[†]　Harutaka ISHIZU[†]
Kenji ONISHI[†]　Hidetoshi TAKASUGI[†]
and　Osamu TATEBE[††]

Recently, it is increasing interests in huge scale storage system which enables to up to exabyte scale. This research paper evaluates the limit of the capacity enhancing in a current storage system. In addition, the design that achieves the storage system of the exabyte class based on Gfarm v2.4 is examined, the prototype system is made actually, and the result of the evaluation is reported.

1. はじめに

近年，ビジネス向け，コンシューマ向けを問わず個人や，システムが利用するデータ量の増加が顕著であり，クラウドコンピューティングの一般化と共に今後益々，ネットワーク上に必要なストレージ容量が大きくなっていくと考えられる．

図 1 に LAN 回線，中継（伝送）回線の帯域，CPU，スパコンの処理能力，ディスクの容量の傾向を示す[1]．図 1 はあくまでこれまでの伸びからの経験則でしかないが 2000 年から 10 年間で概ね 2^10 倍の関係がある．

科学技術分野では，次世代 DNA シーケンサ，加速器，望遠鏡などの実験装置の急速な発達により一日に数百ギガバイト（GB）～数テラバイト（TB）ものデータが生成されるようになった．また，ビジネス面でも，世界中で発信されるウェブページを基にしたデータ解析も盛んに行われているが，ウェブページの総容量は 300 エクサバイト（EB）程度になると報告されている[2]．より身近な例として，日本において 1 年間に出荷される監視カメラ台数およそ 100 万台の映像データを半年蓄積すると仮定すると数エクサバイト（EB）のストレージ容量が必要となる．

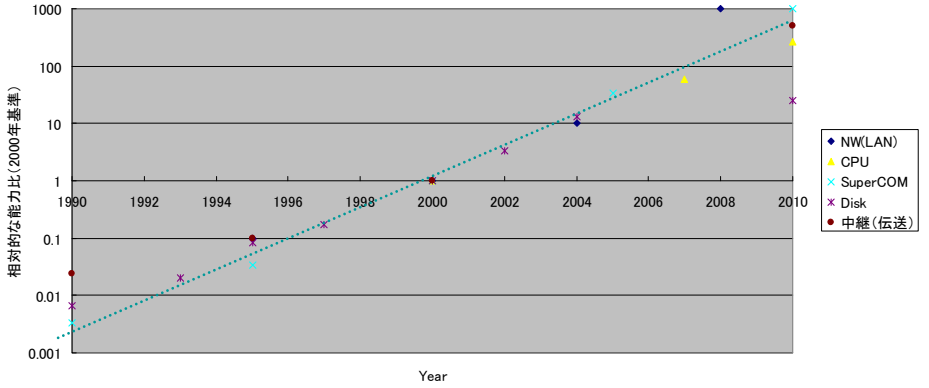

図 1.NW の帯域，CPU の処理能力，Disk の容量の向上（2^10 倍の法則）

本稿では，今後のクラウドにおける大規模ストレージに求められる要件を考察した．

† ＮＴＴコミュニケーションズ（株）
NTT Communications Corporation
†† 国立大学法人筑波大学
University of Tsukuba

また，その実現における設計検討を実施した．更に，実際に Gfarm v2.4 をベースにしたプロトタイピングの作成及び評価を行い，本設計方式の可能性を評価した．

2. クラウドにおける大規模ストレージに求められる要件

2.1 現状のストレージシステムの限界

1 拠点のクラウドストレージでまかなえるデータ量は数 PB～数十 PB 規模であり，この容量を 1,000 倍，10,000 倍に増やすことは難しい．容量を増やすための難しさの原因は大きく二つある．一つ目は物理的な容量の問題，二つ目はシステムソフトウェアの問題である．物理的な容量の問題では，現在 3TB の HDD が利用可能であるが，19 インチラック(42unit)に詰め込み可能な HDD の数は（4unit のサーバで 40 本 HDD が入ると想定）400 本ほど（1.2PB）であり，冗長性を考慮すると（raid6 を 8:2 の構成で組む）1 ラックで 1 PB となる．10PB，100PB の容量を確保するためには HDD だけでも 10 ラック，100 ラックが必要になる．システムソフトウェアについてはファイルデータの数を考慮する必要があることである．例えば，国産の OSS（Open Source Software）として，IPA でも評価の高い[3]Gfarm ファイルシステム[4] [5]を取り上げると，オンメモリにてメタデータを保持する仕様のため 100GB のメモリを搭載したメタデータサーバの場合，100M（1 億）ファイルが限界（メタデータは 1 件当り 1KB 程度）である．1 ファイルあたり平均 10MB とすると 1PB，1GB とすると 100PB が限界となる．また，現在主流のファイルシステムである ext3 は最大 16TB である．スパコンでよく利用される Lustre ファイルシステムでは，ext3 のファイルシステムを理論上，最大 8,150 利用することができるが，それでも 100PB 程度が限界となる．

このように 1 拠点のクラウドストレージでまかなえるデータ量はせいぜい 100PB 規模であるため，今後増え続ける 1,000 倍，10,000 倍のデータを格納するためのストレージの構築には，データセンタを跨いだストレージシステムの構築とそれに対応可能なソフトウェアの拡張が必要となる．

また，詳細は公開されていないが，Google の保持する検索用のデータは全 Web データの内 0.004%保持している[6]との情報と，現状 300 エクサバイト（2009 年時点で 280 エクサバイト）程度 Web 上のコンテンツがあるといわれている[2]ことから，トータルでは 10 ペタバイト程度 Google は自社のデータセンタ内に保持していると想定される．

本節の最後に，Gfarm ファイルシステムの単体方式（現状の方式）の場合に，今後 10 年程度で既存のアーキテクチャを使用し続ける場合の限界容量の試算を図 2 に示す．本試算は，サーバのメモリの容量拡大と，ディスクの容量拡大の両方が限界容量に影響を与えることからシミュレートした結果である．この試算結果については，今後必要となるストレージシステムが数年後に必要な容量と比較した場合に図 2 の示す曲線より上の領域にある場合は，そのアーキテクチャを使い続けることは難しいことを示している．

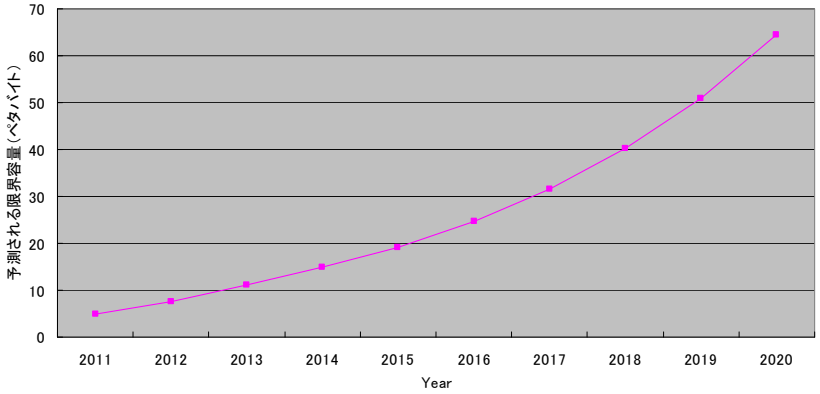

図 2.現時点での単体方式のアーキテクチャの限界容量と今後予想される限界

2.2 Gfarmファイルシステムでの具体的なアーキテクチャ検討

本節では，具体的に Gfarm ファイルシステムを対象にして，エクサバイト級のシステムを実現可能なアーキテクチャを比較検討する．

まず，現状の Gfarm ファイルシステム（単体（ベース）方式）で容量の制限を生み出している原因は前節で明らかにしたとおり，標準的な IA サーバの搭載可能メモリの制限による管理可能なファイル数の限界にある．したがって，大容量に拡張するためには単体での管理可能なファイル数を増やす拡張の方向性と，複数のシステムを連携させて拡張する方向性の二つが考えられる．

前者の方向性からは，メタデータを小さくする方式，メモリを仮想的に大きくする方式が考えられ，後者の方向性からマウントの方式を工夫する方式（2 方式），メタサーバ自体を連携させる方式が考えられる．

メタデータを小さくする方式とは，現状のメタデータの管理方法等を見直し，1KB 程度あるメタデータの容量を小さくする方式（メタデータのコンパクト化方式）である．

メモリを仮想的に大きくする方式とは，全てのメタデータをオンメモリでは持たずに，一部をハードディスクに持つことにより管理可能なメタデータの数量を増やす方式（メタデータのスプール化方式）である．

また，マウントの方式を工夫する方法として一番目は別々の Gfarm ファイルシステムをユーザが並列となる場所にマウントし管理する方式（運用で使い分けをすることにより擬似的に大容量になる：並列マウント方式）であり，二番目は Gfarm ファイルシステムを改造して一つのディレクトリ配下に入れ子に別々の Gfarm ファイルシステムをマウントする方式（そのため，ユーザは余り複数の Gfarm ファイルシステムがあることを意識しなくてすむ：階層マウント方式）である．

最後の方式は複数の Gfarm ファイルシステムを分散して連携させる方式であり，お互いの Gfarm ファイルシステムが透過的に管理下のファイルの位置情報等にアクセスできる工夫をする方式である．

表 1.大容量化可能なアーキテクチャ評価

アーキテクチャ	メリット	デメリット	拡張可能な容量
単体（ベース）方式	--	・容量の限界	10PB
並列マウント方式（複数 NameSpace）	・現状のシステムのまま対応が可能	・利用者側の負担（設計，管理）が大きい	100PB ～200PB
階層マウント方式（階層 NameSpace）	・大容量対応可能	・効果は見込まれるが，設計の変更が膨大 ・他のクラウドとの連携は難しい	1EB ～10EB
メタデータのコンパクト化方式	・メタデータを圧縮した分に比例して容量が増大する ・他の方式と組み合わせればより効果が見込まれる	・単体では効果が少なく，設計の変更が膨大	20PB
メタデータのスプール化（仮想メモリ）方式	・比較的簡単に対応可能（改造の難易度低い）	・パフォーマンスが極端に低下する	100PB
分散連携サーバ（クラスター化）方式	・大容量対応可能 ・利用者が NameSpace を気にする必要がない	・効果は大きいが，設計の変更が必要 ・他のクラウドとの連携も難易度が低い	10EB～

各々の方式を拡張可能と想定される容量と共に整理すると，表 1 の通りになる．将来の大容量化（エクサバイト級）を踏まえた拡張方式としては，分散連携サーバ方式が拡張可能な容量，クラウドとの親和性を考えるとデメリットの少ない一番有力なアーキテクチャであるといえる．

2.3 まとめ

本章にて，クラウドにおける大規模ストレージとして拡張可能なアーキテクチャを Gfarm ファイルシステムに対して比較検討した，その結果分散連携サーバ方式が最適な方式であることが分かった，実際の方式決定に当たっては，図 2 を踏まえて将来どの程度容量の拡張が必要かを想定し，最適なストレージシステムのアーキテクチャを選択する必要がある．

ただし，アーキテクチャの選択については容量の拡張のみでなく，メリット・デメリットを総合的に考慮する必要があり，例えば既存のソフトウェアの変更が許容できない場合は容量に限界があるものの並列マウント方式を選択するケースもありうる．

3. Gfarm v2.4 をベースとしたエクサバイト対応設計概要

3.1 設計方針

ストレージシステムの分散連携を実現するためには，各ストレージシステムを束ねるメタサーバ間の連携をする機能が必要であり，それを実現するのが分散連携サーバとなる．分散連携サーバの設計にあたり，グリッド分野で標準化が進められている Resource Namespace Service (RNS) [7]の仕様を用いる方式（RNS 方式），および Gfarm ファイルシステムのメタサーバを利用する方式（URL 方式）を検討した．RNS は，階層的な名前空間の管理を行い，名前から対応するアドレスに変換するサービスに対するインターフェースの標準である．アドレスは Web Services Addressing Endpoint Reference (EPR)により表現される．EPR は URL を拡張したものであり，XML 文書として表される．EPR として，別の RNS を参照することにより，階層的な名前空間の管理を分散して行うことが可能となる．また，EPR として，クラウドストレージを参照するようにすると，各クラウドストレージを束ねることが可能となる．また，各名前には任意のメタデータを付加することが可能である．

一方，Gfarm ファイルシステムのメタデータサーバでは，シンボリックリンクを保持する機能がある．シンボリックリンクとして，通常の相対パス，絶対パスだけではなく，URL を保持するようにすると，RNS における EPR と同様，別のサービスを参照することが可能となる．分散連携サーバでは，ファイルシステムのようにファイルを保持するわけではないため，ファイルを管理する機能は必要ないが，階層的な名前

空間と,別サービスを参照するためのアドレスの保持が必要であり,その機能は Gfarm ファイルシステムのメタデータサーバにおけるディレクトリ管理とシンボリックリンク管理において実現可能である．従って，Gfarm ファイルシステムのメタデータサーバ機能の内，ディレクトリ管理とシンボリックリンク管理を利用することにより，分散連携サーバの実現が可能となる．今回は，RNS 方式については標準化が遅れていることから，URL 方式による実装方針とした．

URL では，gfarm://server:port/path/name という形式の Gfarm URL を利用することにより，別メタデータサーバを参照する．これにより，分散連携サーバの分散を可能とする．また同一形式の URL により，別の Gfarm ファイルシステムに対する参照ができるようにすることにより，クラウドストレージへの参照も可能とする．なお，Gfarm ファイルシステム以外のクラウドストレージへの参照については，それぞれの URL を用いることとなる．

表 2.ネームスペースの管理方式の比較

	RNS方式	URL方式
概要	Open Grid Forum (OGF) で標準化が行われている方式	現行のGfarmファイルシステムで採用している方式を拡張した方式
ファイル等の位置を示す情報の形式	EPRと呼ばれる，W3Cで標準化されたXML形式のデータを使用	他のクラウドを指し示すことができるように機能拡張したURLを使用

3.2 大規模広域分散ストレージプールの設計

(1) 分散連携サーバの設計について

分散連携サーバは，階層的な名前空間を管理し，クラウドストレージの連携を行うサーバである．分散連携サーバでは，Gfarm ファイルシステムのメタデータサーバ機能を利用し，ディレクトリ管理，シンボリックリンク管理機能を利用して，分散連携サーバを構成する．Gfarm ファイルシステムでは，広域に分散した環境を想定した認証システムが用いられており，これにより，地理的に分散したサーバとしても実現できるようになる．分散連携サーバの一部の障害によりシステム全体にアクセスできなくなることを防ぐために，分散連携サーバが単一障害点とならないように設計する必要があるが,これについては,分散連携サーバを冗長に設置することにより対処する．

分散連携サーバにおける名前空間の変更は少ないため，冗長に設置したサーバ間の同期は緩やかなものでもとくに問題は発生しないと考えられる．利用者が増えた場合の性能スケールアウトについては，分散している分散連携サーバに利用者を分散させてアクセスさせることにより実現可能である．

図 3 に分散連携サーバを Gfarm ファイルシステムのメタデータサーバ（gfmd）を用いて設計した図を示す．分散連携サーバは赤い波線で囲まれた階層的な名前空間全体を管理する．階層的な名前空間は，赤い丸で囲まれたように部分的な階層名前空間で分割され，その部分的な階層名前空間は gfmd で管理される．
各 gfmd ではディレクトリ管理とシンボリックリンク管理を行い，部分的な階層名前空間の管理（部分ディレクトリ管理）と，gfmd 間の参照および gfmd から拠点のクラウドストレージへの参照を管理する．つまり，赤い丸の中（同一 gfmd 内）の赤い矢印はディレクトリによる管理，赤い丸同士（gfmd 間）および赤い丸からクラウドストレージ（gfmd とクラウドストレージの間）への赤い矢印はシンボリックリンクにより管理する．例えば，先頭のルートノードから，com，comp-a，mail とたどる場合（以下/com/comp-a/mail のように記述）では，com は異なる gfmd への参照となっており，以降の名前空間は異なる gfmd で管理される．comp-a も同様である．Mail は企業 A のクラウドストレージへの参照であり，企業 A のクラウドストレージが参照される．このように，分散連携サーバは，階層的な名前空間を複数の gfmd で管理し，複数のクラウドストレージの連携を行う．

(2) （分散連携サーバ対応）クライアントの設計について

分散連携サーバのクライアントの設計にあたり，分散して配置され，シンボリックリンクで連携している分散連携サーバに対し,利用者にはそれを意識させることなく,透過的に gfmd 間およびクラウドストレージへの参照を含むようなパス名(/edu/univ/cs など）を解決する必要がある．
これを実現するため，クライアントライブラリに対し，以下の機能を追加する必要がある．

- 分散連携サーバにおけるパス名解決
- 各種 API の分散連携サーバ対応

分散連携サーバにおけるパス名解決では，パス名解決中にシンボリックリンクがあった場合，そのリンク先を取得し，パス名解決を続行するよう設計を行う．シンボリックリンクが Gfarm URL 形式で参照されている場合も，指定された gfmd に接続を行い，パス名解決を続行するよう設計する．これらのパス名解決においては，シンボリックリンクそのものの情報を返すような API を除き，クライアント API からはサーバ

の分散に対し透過的であるように設計する．

各種 API の分散連携サーバ対応では，gfs_stat API, gfs_realpath API, gfs_rename API, gfs_link API をはじめとする各種クライアント API を，分散連携サーバおよびシンボリックリンク対応にするよう設計する．

gfs_stat API は，指定されたファイル，ディレクトリの情報を返す API である．分散連携サーバ対応では，パス名で指定されたエントリがシンボリックリンクの場合，シンボリックリンクそのものの情報ではなく，シンボリックリンクのリンク先の情報を返すように設計する．シンボリックリンクが Gfarm URL 形式の場合も，指定された gfmd に接続し，リンク先の情報を返すように設計する．

gfs_realpath API は，与えられたパス名を元に，正規の Gfarm URL を返す関数である．分散連携サーバ対応では，与えられたパスが複数の gfmd をたどるような場合も，正しく正規の Gfarm URL を返すよう設計する．ここで，正規の Gfarm URL とは，パス名中のすべての Gfarm URL を含むシンボリックリンク，余分な/や/./，/../の参照を解決したものである．

gfs_rename API は二つのエントリのパス名を引数として取り，エントリの名前を変更，あるいは場所を移動する（rename 操作を行う）API である．gfs_link API は二つのエントリのパス名を引数としており，リンクを作成する（link 操作を行う）API である．これら API の分散連携サーバ対応では，引数でとる二つのエントリのパス名に対し，二つのエントリが異なる gfmd で管理されている場合にそのエントリ間で rename 操作，link 操作を禁止し，エラーを返すよう設計する．

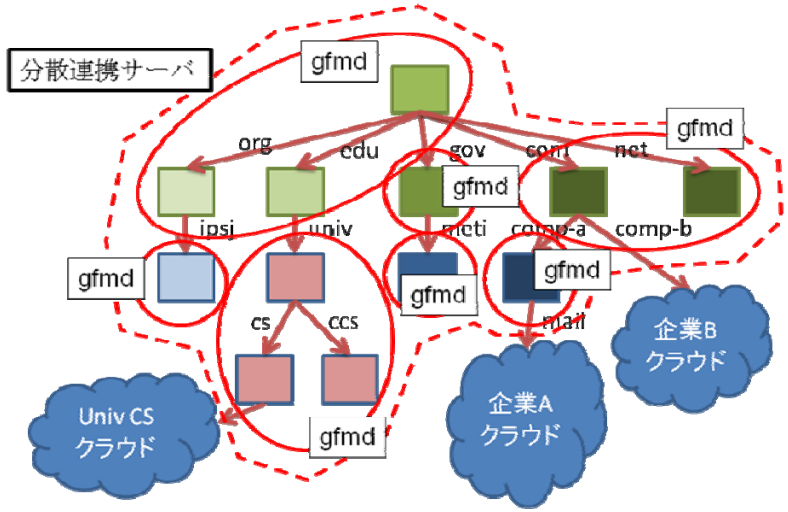

図 3.分散連携サーバの連携の様子

3.3 Gfarm v2.4 をベースとしたプロトタイプ実装について

本節では，具体的に前節で述べた設計を Gfarm v2.4 にて実現させるために行った詳細設計の概要，代表的シーケンスを示す．

以下，作成した設計にて重要な部分であるパス名解決処理のシーケンス例として，パス中にシンボリックリンクが存在する場合を示す．例としては，/dir1/to_dir2/file2 というシンボリックリンクを含むパス名に対し，gfs_stat API を適用する例である（図 4）．

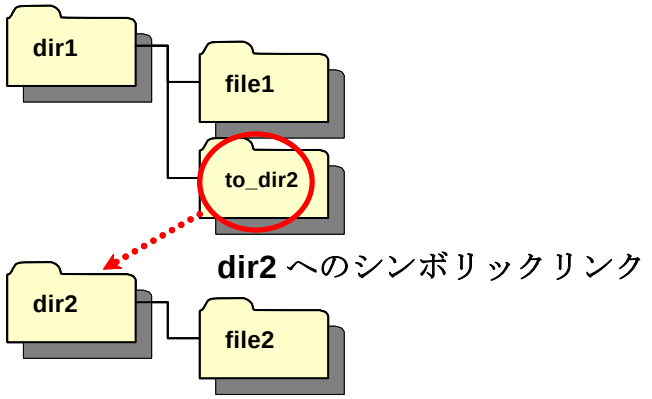

図 4.メタデータサーバで管理するディレクトリ情報例

この場合は，シンボリックリンクが存在するため一回の複合処理のリクエストで，処理が完了しない．そのため，一度目の複合処理のリクエストでシンボリックリンクのリンク先の情報を取得し，再度リクエストを発行する．

gfs_stat(“/dir1/to_dir2/file2”)
→　メタデータサーバからの返答で/dir/to_dir2 はシンボリックリンクであり，リンク先は/dir2 であるとの情報を取得．
gfs_stat(“/dir2/file2”) 相当のリクエストを発行する．
→ 処理完了．

このとき，クライアントからメタデータサーバへのリクエストは図 5 の様になる．

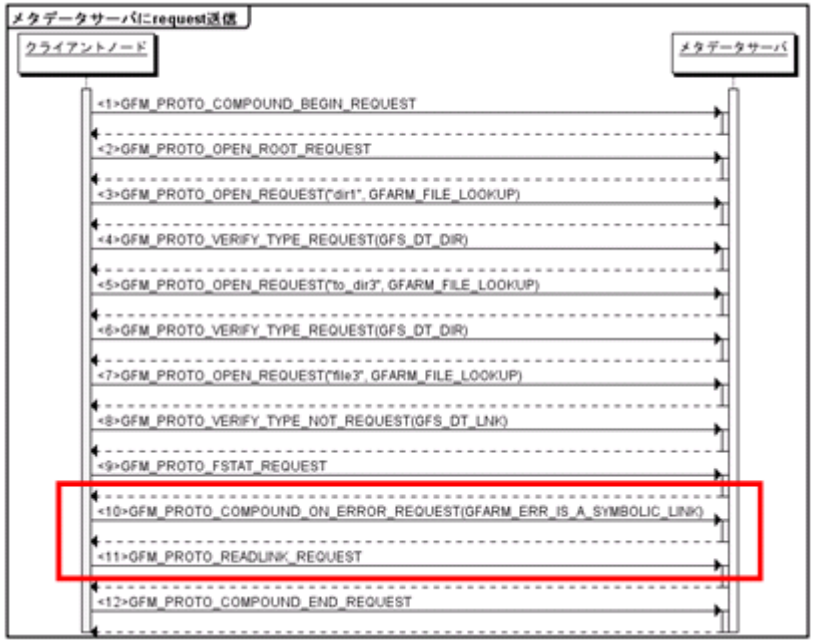

図 5.クライアントからメタデータサーバへのリクエストシーケンス

このリクエストでは，GFM_PROTO_OPEN_ROOT_REQUEST により，ルートディレクトリをオープンした後，dir1，to_dir2，file2 を順次オープンするリクエストを出している．dir1，to_dir2 のオープンでは，それぞれのリクエストごとに，そのエントリのタイプがディレクトリであるかを確認している．その確認に成功すれば次に進むが，確認に失敗した場合は，エラー処理に進む．エラーの内容がディレクトリではなく，シンボリックリンクである場合，GFARM_ERR_IS_A_SYMBOLIC_LINK というエラーが返り，GFM_PROTO_COMPOUND_ON_ERROR_REQUEST(GFARM_ERR_IS_A_SYMBOLIC_LINK)に進む．このとき GFM_PROTO_READLINK_REQUEST が実行される．

これにより，パス名解決の途中でシンボリックリンクにぶつかった場合，そのリンク先の情報をクライアントに返すことが可能となる．その場合，クライアントライブラリでは，リンク先のパス名を利用してさらにパス名解決のリクエストをメタデータサーバに発行することにより，全体のパス名解決を図る．

4. プロトタイプシステムの検証とその評価結果

4.1 評価条件について

プロトタイプシステムを評価するにあたって，元となる Gfarm ファイルシステムと比較して各動作が機能，性能の両面から同等なことを重視して実施した．

また，商用レベルでの将来的な活用も想定し，異常系の動作についても問題なく使用可能か確認する．

- WAN 環境を想定した試験環境で，分散連携サーバ機能を追加した場合の基本的な機能が問題なく動作すること．
- 各サーバが障害を起こした場合にエラー処理などが問題なく動作すること．
- IOZone を用いた性能試験にて，分散連携サーバ機能を追加した場合にも，従来（ベースとなる Gfarm v2.4）と同様の性能がでること．

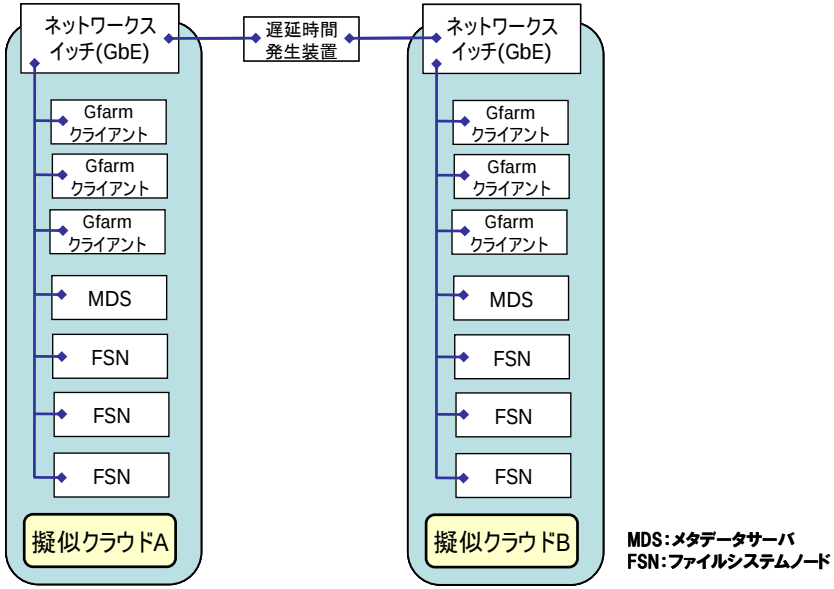

図 6.プロトタイプシステムの評価環境図

4.2 機能試験結果（異常系試験含む）について

機能確認試験では，以下の試験項目をプロトタイプシステムに対して行った．

- ファイルアクセス試験（読み書き，書き込み，変更，削除等）
- ディレクトリアクセス試験（ディレクトリ作成，変更，削除，ディレクトリ配下にファイル作成等）
- シンボリックリンク，ハードリンク操作試験
- 権限試験（権限設定，変更を行い，設定通りのアクセス権になっているか確認）
- 異常系試験（リンク先の Gfarm ファイルシステムのメタデータサーバ，ファイルシステムノードの障害等）

ファイルアクセス/ディレクトリアクセス試験において，問題点はなかった．シンボリックリンク試験では，リンク先の Gfarm ファイルシステム上へ，ln コマンドによるシンボリックリンク作成はできなかったが，gfln コマンドでのシンボリックリンク作成が可能であることを確認した．また，リンク先の Gfarm ファイルシステムへのシンボリックリンクのコピーや移動ができなかった．権限試験においては，Other ユーザにおいて，ファイルのリネーム及び，削除が可能であった．

異常系試験においては，ファイル作成中にリンク先のメタデータサーバまたはリンク先のファイルシステムノードがダウンした場合，ファイルは作成されるが，データが書き込まれないことを確認した.その他の異常試験項目では問題は見られなかった．

したがって，プロトタイプ実装であることを考慮するとほぼ想定どおりの動作をしていることになり，商用化に向けた軽微な修正をすれば充分実用に足りると考えられる．

4.3 性能試験結果について

性能確認試験では，以下の項目をプロトタイプシステムに対して行った．

- メタデータサーバリンク機能を使用しない場合の性能を測定．（IOZone を使用）
- メタデータサーバリンク機能を使用した場合の性能を測定．（IOZone を使用）

試験結果は，図 7,8 に示す通りである．メタデータサーバリンク機能を使用しない場合，使用する場合と比べると，ファイルサイズにより，メタデータサーバリンク機能による影響が異なる事が確認できた．

具体的にはファイルサイズが 100KB 以下のような小さなファイルの場合は大きな性能差は見られないが，ファイルサイズが 1MB を越えたあたりから，性能差が見られ始め，10MB 以上ではメタデータサーバリンク機能がある場合に，スループットがおよそ 1/8 になった．

ただし，試験構成上，遅延装置をクラウド間においているため（図 6），10MB 以上の大きなファイルサイズだと，遅延装置の向こう側にあるデータに対して読み書きする時に，より影響を受け易く性能低下してしまうためといえる．

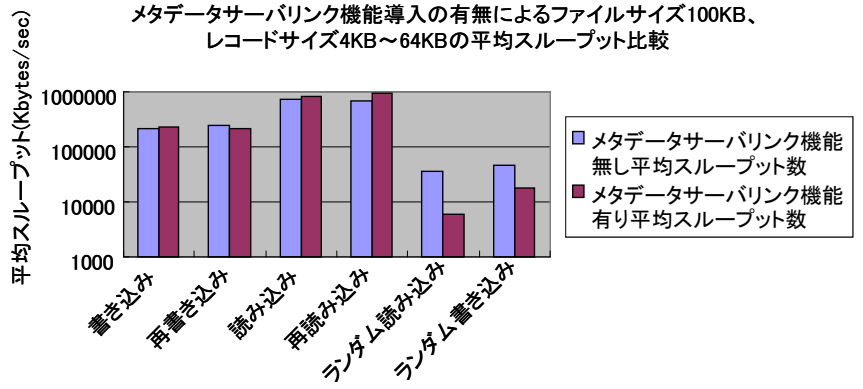

図 7.ファイルサイズが小さい（100KB）場合の性能試験結果

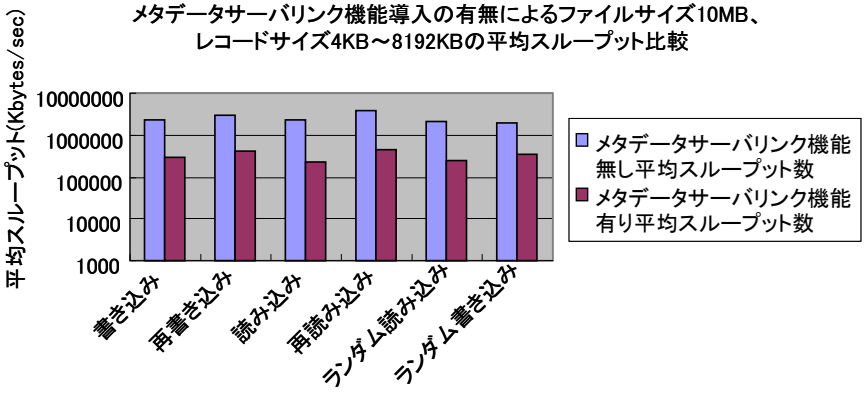

図 8.ファイルサイズが大きい（10MB）場合の性能試験結果

5. むすび

本研究では，現状のストレージシステムの限界について考察を行い，エクサバイト級のシステムに対して Gfarm ファイルシステムをベースにした場合の拡張方式を比較し，分散連携サーバ方式が優れていることを示した．また，実際に分散連携サーバ方式による設計を行った．設計にあたり，利用者が連携しているクラウドストレージを透過的にアクセスできることを考慮した．また，追加したクラウドストレージに応じて，利用者が増えることを考慮して，利用者あたりの性能が落ちることなく，ストレージ全体の性能はスケールアウトして向上していくよう考慮した．

更に，Gfarm v2.4 をベースにしたプロトタイプの実装，評価を行った．評価においては，商用環境での利用に耐えうる

かを重視し，異常系の試験及び性能試験についても行い，本方式にて商用へ適用できる可能性が充分高いことを示した．

今後はクラウドストレージの拠点数を増やしながら動作の確認，および性能評価を行い，実際に1,000拠点，10,000拠点まで連携可能なように設計にフィードバックしていく予定である．また，負荷試験により，商用のシステムとして利用することができるよう品質を高めていく予定である．

6. 謝辞

本研究は，平成 22 年度産業技術研究開発委託費（次世代高信頼・省エネ型 IT 基盤技術開発事業（「クラウドコンピューティングのアカウンタビリティを向上させる研究・開発事業」[8]の成果を含む．

参考文献

1) http://museum.ipsj.or.jp/computer/super/index.html
2) http://internet.watch.impress.co.jp/docs/news/20090615_294083.html
3) IPA ：http://www.ipa.go.jp/about/press/20101027.html
4) 建部修見, 曽田哲之, 関口智嗣: 広域仮想ファイルシステム Gfarm v2 の設計と実装, 情報処理学会研究報告, 2004-HPC-99, pp. 145−150 (2004).
5) 建部修見, 曽田哲之: 広域分散ファイルシステム Gfarm v2 の実装と評価, 情報処理学会研究報告，2007-HPC-113， pp.7-12 (2007).
6) http://theroxor.com/2010/10/28/the-awesome-size-of-the-internet-infographic/
7) http://www.ogf.org/documents/GFD.171.pdf
8) 平成 22 年度経済産業省 平成 22 年度産業技術研究開発委託費（次世代高信頼・省エネ型 IT 基盤技術開発事業）クラウドコンピューティングのアカウンタビリティを向上させる研究・開発事業 事業報告書（2011）：http://www.meti.go.jp/policy/mono_info_service/joho/cloud/2010/index.html