Vol.2011-HPC-130 No.36
2011/7/28

耐障害性を高めた分散ストレージシステムの開発とその評価

石津晴崇[†]　永岡孝[†]　大西健司[†]
高杉英利[†]　建部修見[††]

近年様々なサービスが生み出されており，特にクラウドサービスが注目を集めている．クラウドサービスを開始するにあたっては需要予測が困難なこともあり，必要とされるストレージの投資計画が困難な場合がある．そこで，オープンソースのソフトウェアである Gfarm に着目し，サービスの基盤システムとして利用するために必要な機能を新たに開発することにより，安価に小容量から大容量までスケールアウト可能なストレージシステムを開発した．このシステムの評価を行った結果，機能面では，システム内のシングルポイントを無くし，自律的にファイルの複製状態を維持管理することで耐障害性を高め，ディザスタリカバリにも利用できることを示す．性能面では，オープンソース版 Gfarm と比較して同等であることを明らかにする．さらに，市販アプリケーションと組み合わせて評価を行い，商用サービスに利用可能であることを示す．

Development and Evaluation of a Distributed Storage System Improved Fault Resistance

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

Various services are provided in recent years, and especially cloud services attract increasing attention from users. Because it is difficult to forcast the demand for cloud services, it is also difficult to make the investment plan of the storage system. Then, we picked out "Gfarm" which is one of open source software, and developed the scale-out storage system based on "Gfarm", which is enable to built storage system from small capacity to large capacity at a low cost. This paper describes how to improve fault resistance, for instance, removing a single point of failure in the system, keeping the number of replica files autonomously. These are enable user to use this for disaster recovery. And also this paper shows that developed system provides almost the same performance as the original Gfarm. In addition, we evaluate the combination this developed system and commercial application for the cloud service.

1. はじめに

近年，企業や個人が作成する情報の電子化が進みデータ量が飛躍的に増加している．電子データの増加に伴い，それらを保存するためのストレージには大容量化が求められている．しかしながら，数百テラバイト程度の大容量であればバイト単価が安いストレージ製品も存在するが，小容量の規模では販売していないため初期投資額が大きくなってしまう問題がある．また，クラウドサービスは需要予測が不透明であり,ストレージの投資計画が困難な場合が多い．そのため小容量から始めて必要に応じて最適な製品でスケールアウトし，小容量でも安価なストレージが望まれている．

一方、IA サーバの性能向上，低価格化が一層進んでおり、コストパフォーマンスが非常に良い製品が多く，サーバにも関わらず数テラバイトの容量を保存可能な製品も存在する．このような製品を利用してファイルサーバとすると，必要な時に必要な製品を購入することができるが，台数が増えていくため将来的には管理と利用方法が複雑になってしまう問題が発生する．

このように，ストレージ製品が持つ小容量から始めて必要に応じて最適な製品で容量を増加できないという課題と，中程度の容量を持つサーバは逐次増やせるが管理と利用方法が煩雑という課題の両者を解決する手法の一つとして，分散ストレージシステムがある．このシステムは複数のサーバをネットワーク越しに一つのストレージとして管理し，利用することができる．

このような分散ストレージシステムは利用方法や目的に応じて，構成するサーバへの機能分担，データの分散方式，サーバの管理方式，規模などの点を中心に設計されており，Gfarm[1][2][3]，lustre[4]，Cassandra[5]などがオープンソースソフトウェアとして公開されている．

我々は，オープンソースソフトウェアとして公開され，汎用的なインターフェイスで利用可能な Gfarm に着目したが，サービスの基盤システムとして利用する場合，いくつかの改良すべき項目があった．そこでサービスの基盤システムとして利用するために必要な機能を追加し，安心・安全に利用するためのストレージシステムを開発した．

本稿の構成は以下の通りである．2 章でストレージシステムに求められる要求条件と既存のソフトウェアが抱えている課題を説明する．3 章で今回開発したシステムの特徴を説明する．4 章でストレージシステム単体での評価結果と他アプリケーションと連携した場合の評価結果について説明する．5 章で今後の課題についてまとめる．

[†] NTT コミュニケーションズ株式会社
NTT Communications Corporation
[††] 国立大学法人筑波大学
University of Tsukuba

2. ストレージシステムに求められる要求条件と既存システムの課題

2.1 ストレージシステムに求められる要求条件

ストレージシステムには企業活動に重要な情報がデータとして保存されるため消えてなくならないことがまず求められる．現在の多くのシステムでは，保存媒体であるハードディスクの障害に備えて RAID を導入している．他のストレージシステムの場合も特別な用途の場合をのぞき，消えてなくならないことが求められる．

ストレージに保存されるデータは，利用者が意識的に削除しない限り増加の一途をたどるため，将来の需要予測による投資ではなく，実際に利用する容量分のみの投資をし，容量増加に対応できることが求められる．ストレージへの要求容量と実際の使用容量を分離する機能としてシンプロビジョニングがあるが，この機能を有するストレージ製品は高額なためコストが増加してしまうという問題がある．

ストレージシステムを実際に導入する際には，データが消失しないこと，コスト，容量の他にも要求される条件が存在する．他の要因の例としては，SI 企業などがサーバやストレージを自社や他社へ導入する場合，構築を行うメンバーと監視，運用を行うメンバーが異なることが多く，運用，監視メンバーが容易に該当システムを扱えることが挙げられる．このように，ストレージの機能そのものとは直接関係しないとみられる運用に関することも要求される．

様々なシステム毎に求められる要求は異なるが，共通基盤として利用することを考慮すると上記要求条件に加えて，第一に，ストレージシステムを構成する部位に障害が発生した際にストレージサービスが長時間停止しないために，シングルポイントがないことが条件である．第二に，導入の際に複数の製品を組み合わせてシステムを構築することが多いため，組み合わせ時にソフトウェアに改良ができるだけ発生しないように，汎用的なインターフェイスで利用可能なことが求められる．

このように，基盤システムで利用可能なストレージシステムについては，データがなくならないこと，安価であること，運用し易いこと，シングルポイントがないこと，汎用インターフェイスで利用可能なことが求められる．

2.2 既存システムの課題

前節で示したストレージシステムへの要求を満たすために，我々は分散ストレージシステムに着目した．分散ストレージシステムは，複数のサーバをネットワーク越しに一つのストレージとして管理し，利用することができる．そのため，サーバ単体でハードディスク障害を回避しつつ，小容量からスケールアウトして安価に大容量が実現可能である．そこで我々は，汎用的なインターフェイスも利用可能で，小容量でも安価に分散ストレージシステムが構築可能なオープンソースソフトウェアの Gfarm に着目した．本節ではオープンソース版 Gfarm の概要とオープンソース版 Gfarm で対応できない要件について説明する．

2.2.1 オープンソース版 Gfarm の概要と対応できている要求条件

(1) 動作概要

Gfarm は，大規模な PC クラスタで必要とされる大容量，高性能なファイルシステムを実現することを目的として，2001 年にオープンソースとして公開され，2007 年に現在のメジャーバージョンとなるバージョン 2 系が公開されており，開発完了時は 2.4.0 であったが，現在は 2.4.2 がリリースされているソフトウェアである．

Gfarm は，ファイルの管理情報などを集中管理しているメタデータサーバ，ファイルの実体を保持しているノードサーバ，Gfarm を利用するクライアントサーバから構成され，それらにはライブラリ，各サーバ向けのプログラム，管理コマンドなどが含まれている．これら 3 種類のサーバはネットワークを介して連携をしており，その連携内容を時系列に沿って記述した動作概要を図 1 に示す．

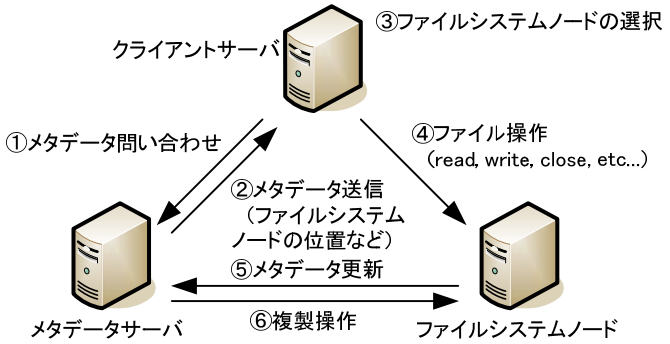

図 1 Gfarm の時系列に沿った動作概要

(2) コストと容量

Gfarm は，性能，ストレージ容量，最大ファイル数などの要件にもよるが，3 種類のサーバを 1 台の物理サーバ上で兼ねることも可能であるため，数百万個のファイル，数テラバイトの小容量であればサーバ 1 台でも構築可能であるため，初期投資を抑えることが可能である．ただし，将来的に容量が足りなくなることを考慮すると，最小構成は 2 台となるが，メタデータサーバは分けることが望ましい． また，ファイルシステムノードは実ファイルの操作，複製処理のみをするため，高性能な製品である必要はなく，安価な製品でコストを抑えることが可能である．

将来容量が足りなくなり，容量増設を行う場合，ファイルシステムノード数に制限がないため，数台の小規模から，数百台の大規模までスケールアウト可能である．さらに．ファイルシステムノードを増設した際に，クライアントノードで Gfarm ファイルシステムを再マウントする必要がないため容易に容量変更が可能である．

(3) サーバ障害によるデータ消失回避

Gfarm は保存するデータの実体ファイルをファイルシステムノードに保存するため，複数のファイスシステムノードを管理している場合，Gfarm に保存したデータの実体ファイルはいずれか一つのファイルシステムノードにのみ保存される．そのため，ファイルシステムノードに物理的な障害やネットワーク障害が発生するとファイルにアクセスできなくなる上，ハードディスクに起因する障害であれば保存したデータの消失の危険性が非常に高い．しかし，Gfarm には手動で複製を作成するコマンドが提供されているため，Gfarm の利用ユーザが別途手動でコマンドを実行すればファイルシステムノードの障害を条件付きではあるが回避することが可能である．また，ファイルシステムノードの物理的な設置条件が IP で通信可能であれば良いことであるため，同一データセンタに限らず遠隔地などの異なるデータセンタにサーバを配置することが可能である．

(4) 汎用 IF での利用

Gfarm はすでに紹介したとおり，3 種類のサーバが互いに連携してファイルを保存する．サーバ間の連携時は専用のプロトコルで通信を行うため，ファイルの読み書きを行うための専用コマンド群が提供されている．しかしながら，FUSE を介して Gfarm をローカルファイルシステムの様にマウントして利用することも可能である．そのため，既存のアプリケーションから専用コマンドを利用することなく直接 Gfarm にアクセス可能である．

2.2.2 オープンソース版 Gfarm を基盤システムとして利用する場合の課題

(1) シングルポイントの存在

Gfarm は保存されるファイルすべてのメタデータ（ディレクトリ情報、ファイル情報，ファイル格納場所等）をメタデータサーバで集中管理する．このメタデータはメタデータサーバのメインメモリ上に展開されており，更新される際は情報の永続化のために PostgreSQL を利用したバックエンドデータベースも更新される．このメタデータサーバを構成するアプリケーション自体には冗長構成等が実装されていないため，メタデータサーバに物理的な障害やネットワーク障害など何らかの障害が発生すると，Gfarm の動作が停止してしまうシングルポイントになっている．また，メタデータサーバを設置しているデータセンタに障害が発生してしまった場合もサービスが停止してしまう．このメタデータサーバはファイルの管理情報等の重要性の高い情報を管理しているため，障害が発生してもサービスが継続できるように冗長化することが必須である．

(2) ファイルシステムノードの選択方法

Gfarm にファイルを保存する際，実体ファイルの保存先ファイルシステムノードは，ファイルシステムノードの CPU 使用率，応答時間などを考慮した上で、クライアントサーバでランダムに選択される．そのため，特定のファイルシステムノードに集中する可能性が低いなどのメリットもあるが，データセンタやラックなどの物理的な条件に基づいて明示的に使い分けることができない課題があるため，ディザスタリカバリ目的での利用に向いていない．そのため，災害等によるデータセンタの障害が発生するとファイル消失を引き起こす可能性がある．そこでこの障害からデータ消失を回避するためには，ファイルシステムノードを明確に指定できる必要がある．

(3) 保存ファイルの手動複製

ファイルシステムノードの障害によるファイル消失を回避するために Gfarm は手動で複製を作成できるコマンドが提供されていることを前節で紹介したが，利用者が個別に複製を作成する必要があることは，良い面もあるが，基盤ストレージシステムとしては課題である．基盤ストレージシステムとして必要な要求条件にデータがなくならないことが含まれることからも，システム内の想定される障害であるファイルシステムノードの物理的な障害やネットワーク障害に対して回避策を講じ，耐障害性を高めることは必須である．すなわち，ファイルが保存される際に，基盤ストレージシステム内で自動的に複製を作成する必要がある．

(4) 運用中の複製確認方法

システム運用を開始するとサーバの物理的な障害やネットワークの断線などの想定される障害に加え，多重障害や他アプリケーションとの連携時特有の障害など何らかの想定外の事象が起こる可能性がある．このような想定外の事象が原因で複製が作成されない場合，状況が把握できず結果的に放置することになり次の障害発生時にデータが消失する可能性がある．すなわち，自動的に複製状態を確認できないという課題がある．

さらに，システムを長期間運用すると故障の発生によるハードウェアの交換が必要になる．すべてのファイルの複製が存在したとしても，故障が発生すると複製がなくなるファイルができ，手動で再度複製を作成するまで複製ファイルがない状態が継続してしまうという課題がある．これらの課題解決のために複製状態を確認し，状態に応じて複製を作成することで耐障害性を高める必要がある．

3. 新システムの開発内容

3.1 新システムの設計内容

2.2.2 節でオープンソース版 Gfarm を基盤ストレージシステムとして利用する場合の課題を明らかにした．これらの課題を解決するための機能について紹介する．

(1) シングルポイント回避のためのメタデータサーバ冗長化

メタデータサーバは Gfarm が管理しているファイルに関するメタデータを保持する中核のサーバであるが，アプリケーションとしては冗長化されていないため，メタデータサーバを冗長化させることで，シングルポイントをなくす．

サーバの冗長化の方式としてよく知られているのはフォルトトレラント型サーバ

の導入，予備機への切り替え，仮想化技術の導入などがあるが，1 台のメタデータサーバで扱えるファイル数の上限，障害発生時のダウンタイム，コストを重要視した結果，現用系から待機系への切り替わり時に発生するダウンタイムを許容することで予備機への切り替え方式を採用することとし，オープンソースソフトウェアの Heartbeat を採用する．

次に，逐次更新される PostgreSQL データの共有方式についてであるが，FC 接続や SAS 接続の共有ストレージを導入することが一般的で高速であるが，導入コストを抑えるためにネットワーク越しにハードディスクのミラーリングが可能なオープンソースソフトウェアである DRBD を採用し，メタデータサーバ切り替わり時にメタデータを引き継ぐこととする．

最後に，メタデータサーバを設置するデータセンタに障害が発生した場合に同一データセンタ内の切り替わり時間よりも長いダウンタイムを許容することで，障害発生前のデータを引き継ぎ，ディザスタリカバリのために遠隔地でサービスが開始できるように，PostgreSQL のデータをバックアップするために Slony-I を採用する．

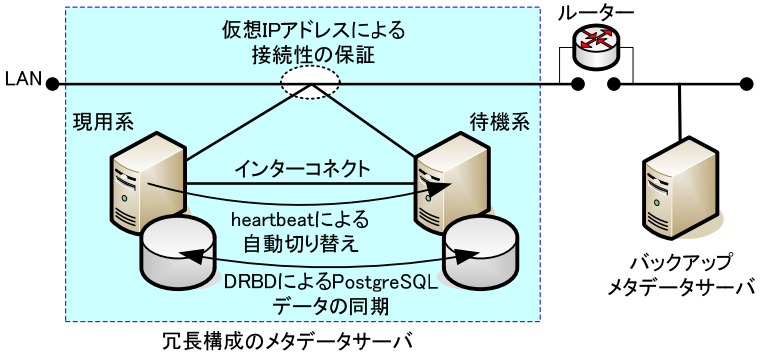

図 2 メタデータサーバの冗長化構成概要

これらオープンソースソフトウェアを導入することでオープンソース版の Gfarm を変更することなくメタデータサーバの物理的な障害，ネットワーク障害だけでなくデータセンタ障害からもダウンタイムが発生するが，サービスが提供できるように耐障害性を高める方式を採用する．本システムで採用したメタデータサーバの冗長化構成の概要図を図 2 に示す．

(2) 保存先ファイルシステムノードの指定機能

オープンソース版の Gfarm はファイルを作成する際，一定の条件の下で正常なファイルシステムノードがランダムに選択される．そのため，ディザスタリカバリを目的に導入しても保存されたファイルによっては同一ラックに複製が作成され，ディザスタリカバリの目的を達成できない可能性がある．

ファイルシステムノードを指定する機能を追加するにあたり，どのファイルシステムノードに実体ファイルを転送するのかを決定するのはクライアントノードであるためクライアントノードにある gfarm2fs を変更し，ファイルシステムノードに優先度を設定することが考えられるが，この方式ではファイルシステムノードの容量が限界に達した際の設定の変更の煩雑さ，ファイルシステムノードのストレージ使用率に偏りが発生してしまうなどの弊害が想定される．そのため，ファイルシステムノードにグループの概念を導入し，グループに対応した新たな関数を定義することで Gfarm のユーザ単位で個別にグループを指定可能な方式とし，柔軟にグループを使い分けることができるようにする．

ファイルシステムノードのグルーピングを Gfarm の利用者が自由に変更できるようにしてしまうと，利用者の初期設定が煩雑になることに加え，ファイルシステムノードを追加した際などにも最適なグループの設定で運用をしていくことが困難になるためファイルシステムノードのグルーピングは運用者のみが設定できる項目とする．

従来の方式は利用可能なファイルシステムノードの一覧を配布していたのに対して，本開発で採用した新方式ではユーザ毎に利用可能なグループに所属するファイルシステムノードの一覧を配布させることで，グループ内，グループ間での柔軟な複製を作成可能にするとともに，従来のランダムにファイルシステムノードを選択することによるストレージ使用率の平準化も可能とする．

これらの方式を採用することで，ファイルシステムノードの物理的な障害，ネットワーク障害，データセンタ障害からもファイル消失を防ぐことが可能になるように耐障害性を高める．同時にディザスタリカバリも可能にする．この新方式の時系列に沿った動作概要を図 3 に示す．なお，図中の点線が従来方式と変更がない箇所を示し，実線が新方式の箇所を示す．

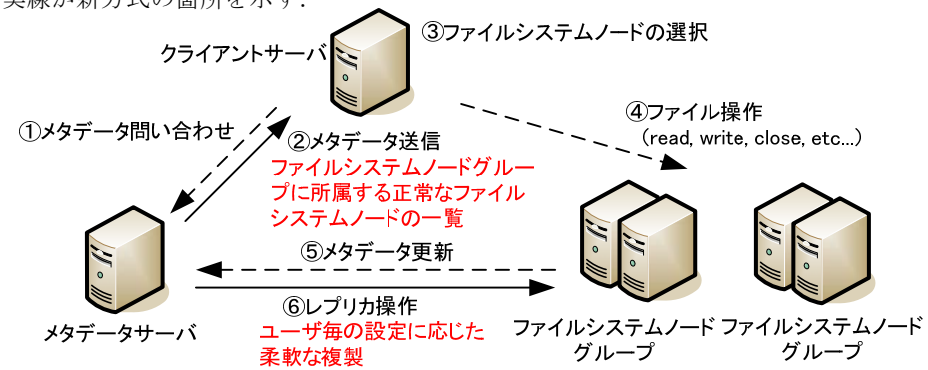

図 3 新方式の動作概要

(3) 自動複製ファイル作成機能

オープンソース版の Gfarm ではファイルを作成する際，クライアントノードが正常なファイルシステムノードをランダムに 1 つ選択して実体ファイルを転送する．そのため､耐障害性を向上させるためには利用者が別途手動で複製を作成する必要があり，手動で実行されるまでの間隔を短くする必要があった上，利用者の運用性を向上させる必要がある．

利用者が意識することなく複数のファイルシステムノードに実体ファイルを書き込む方式には大きく分けて 2 種類ある．1 つ目は，複数のファイルシステムノードに同時に書き込み，書き込んだすべてのファイルシステムノードから書き込み完了の応答が返ってくるまで利用者側のアプリケーションに書き込み完了応答を返さない同期方式である． 2 つ目は，1 つ目のファイルシステムノードに書き込みが完了した直後に利用者側のアプリケーションに書き込み完了応答を返し，並行して二つ目以上のファイルシステムノードに複製する非同期方式である．

本システムは，同期方式は利用者へのレスポンスが低下することを考慮して採用を見送り，非同期方式を採用することで，実体ファイルが 1 ファイルシステムノードにしかない時間を最小限にし，ファイルシステムノードに関係する障害発生時にデータが消失することがないように耐障害性を向上させる．本システムのファイル書き込み時の動作概要を図 4 に示す．なお，図中の点線が従来方式と変更がない箇所を示し，実線が新方式の箇所を示す．

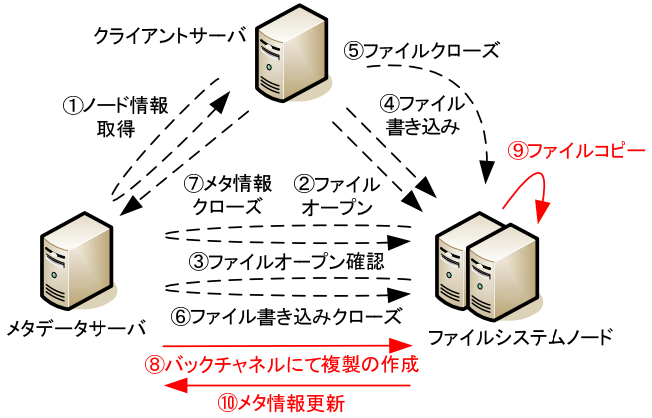

図 4 非同期書き込み方式動作概要

(4) 運用中の複製ファイル自己修復機能

分散ストレージに限らずどのようなシステムでも長期間運用していると物理的な障害，ネットワーク障害やハードウェアの更改が発生する．ファイルシステムノードに障害が発生すると，そのファイルシステムノード保存された実データは消失するため，複製が存在しないファイルができる．

ファイルの複製確認は，(2)で導入した方式により，Gfarm のユーザ，ファイルシステムノードグループの両方に対応させる必要がある．また，運用中にリアルタイムに複製の有無を確認することは困難であるため，定期的に過不足が生じているかを確認し，必要な場合にのみ複製を作成することで余分な負荷を減らすとともに運用者に頼ることなくシステム内で自律的に複製を維持することとする．そのため，ファイルシステムノードのグルーピングに対応した複製確認，作成コマンドを新規に作成し，cron で定期的に実行することでファイルシステムノードの各種障害に対する耐障害性を向上させる．また，複製状態が保存されているメタデータへの操作だけでは障害の影響で実ファイルが作成されない事態が発生した場合を検出できないため，実ファイルの存在確認も可能とする．

3.2 新機能評価要件

3.1 節で示した機能をオープンソース版 Gfarm に追加することが，従来の課題を解決し，耐障害性が向上することを示すための評価手法を述べる．3.1 節(1)項はメタデータサーバの障害に対する追加機能，3.1 節(2)～(4)項はファイルシステムノードの障害に対する追加機能である．特に，ファイルシステムノードに対する追加機能は各機能が組み合わさっているため，機能毎ではなく，サーバの種類毎に耐障害性が向上しているかどうかを評価する．

(1) メタデータサーバの耐障害性評価要件

冗長構成をとったメタデータサーバに対して，プロセス障害，ハードウェア障害，ネットワーク障害を発生させた後で，正常にサービスが再開できるかどうかを確認し，シングルポイントではなくなったかどうかを評価する．

アクティブスタンバイ構成による冗長化では，サーバの切り替えの際にサービスの中断が発生するが，この切り替え時間は 3 分以内に収まることが望ましい．そこで，Heartbeat の実際の切り替わり時間を測定し，3 分以内にサービスが再開できるかどうかを評価する．ディザスタリカバリを可能にする Slony-I による非同期バックアップからのリカバリについては，オペレーションが発生するためサービスが再開できるかどうか機能面のみを評価する．

(2) ファイルシステムノードの耐障害性評価要件

ファイルシステムノードのグルーピングにより，ユーザ毎に指定したグループに所属するファイルシステムノードにのみファイルが保存され，保存されたファイルが自動的に複製されることを確認し，実ファイルが保存されているファイルシステムノードに対してプロセス障害，ハードウェア障害，ネットワーク障害を発生させたあとで保存ファイルが消失していないかを確認し，障害がサービスに影響を与えるかどうかを評価する．

次に，複数のファイルが複製ある状態から一部ファイルの複製を削除し，運用中に

想定される障害によって複製がなくなった状況を発生させ，一定時間経過後にすべてのファイルの複製が正常に作成されることを確認し，障害に対する予防効果があるかどうかを評価する．

(3) 性能評価要件

本システムの読み書き性能を，開発時後に正式版としてリリースされたオープンソース版 Gfarm2.4.0 と比較し，本システムの機能追加により性能が同等であるかどうかを評価する．ただし，1 クライアントからの単純な比較とし，多数クライアントからの同時アクセス時の性能は対象としない．

4. 評価結果

今回開発したシステムのストレージシステムとしての基本的な評価し，市販のソフトウェアと組み合わせた場合の評価を行う．評価方法は通常の利用シーンを想定し，通常のストレージと同様にマウントしてから Linux の標準的コマンドである dd を用いて基本的な性能評価を行う．また組み合わせ試験の際も市販アプリケーションに変更を加えないようにマウントしてから試験を実施する．

4.1 ストレージシステムとしての基本性能評価

今回開発したシステムの耐障害性と基本性能を評価するための測定環境を表 1 に示し，ネットワーク構成を図 5 **エラー! 参照元が見つかりません。**に示す．なお，ファイルシステムノードは性能が異なるサーバを各 2 種類ずつ混在させている．

表 1 評価環境

	メタデータサーバ	ファイルシステムノード			クライアントサーバ
CPU	Xeon E5540 Quad Core	Xeon 3070 Dual Core	Xeon E5540 Quad Core	Xeon E5540 Quad Core	Xeon E5540 Quad Core
メモリ	48GB	2GB	12GB	36GB	36GB
HDD	SAS 15krpm RAID1+0	SATA 7.2krpm RAID6	SATA 7.2krpm RAID6	SAS 10krpm RAID1	SAS 15krpm RAID1+0
NIC	1000BASE-T	1000BASE-T	1000BASE-T	1000BAE-T	1000BASE-T

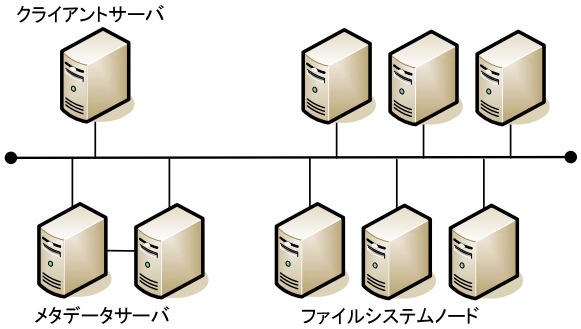

図 5 評価環境のネットワーク構成

(1) メタデータサーバの耐障害性評価

冗長構成をとったメタデータサーバのプライマリ機で監視しているプロセス，ネットワーク，ハードウェアに障害が発生すると，Heartbeat の機能によりスタンバイ機にフェイルオーバーし，サービスが一時中断されるが手動でのオペレーションをすることなくサービスが継続された．よって，機能面から評価するとストレージシステム内に存在していたシングルポイントは解消され，メタデータサーバの耐障害性がオープンソース版 Gfarm と比較して向上している．

各障害に対する詳細な影響については紙面の都合から記述しないが，一例としてプロセス障害の場合を示す．クライアントサーバからのファイルの書き込み中にプロセス障害により切り替えが発生すると，ファイル実体はクライアントサーバからファイルシステムノードに直接転送しているため一見正常に終了するが，メタデータの更新に失敗するため実体ファイルの存在しないファイルとして扱われる．そのため，stat や gfstat などを利用してファイルの状態を確認し，実体ファイルが存在することを含めて正常に書き込みが終了したかどうかを確認することが望ましい．

メタデータサーバの切り替え時間は， heartbeat のログ情報から算出すると，切り替わり開始から再開するまでの時間は平均 7 秒掛かる．なお，この時のメタデータサーバのプロセスの使用メモリサイズは約 242MB であり，サイズが大きくなると切り替え時間も延びる．また，サービス停止時間はこの時間に加えて障害検出までの時間が掛かる．

(2) ファイルシステムノードの耐障害性評価

6 台登録してあるファイルシステムノードの内，6 台をグルーピングした時，テストユーザーが新規ファイルを 20,000 個保存すると，ファイルが複製を含めてグループ内のファイルシステムノードにのみ保存される．よって機能面から評価すると，ストレージシステム内で自動的に複製を作成し，保存先を使い分けられているためファイルシステムノード単位や，ファイルシステムノードグループ単位での各種障害に対す

る耐障害性が向上している．

メタデータサーバの耐障害性評価と同様に，各種障害に対する詳細な影響は省略するが，代表的な障害としてネットワーク障害の場合を示す．ファイルを繰り返し読み出している途中に，ファイルシステムノードでネットワーク障害が発生すると，クライアントでは平均 15 分 38 秒後に読み出しを再開する．一方、書き込み中のファイルシステムノードでネットワーク障害が発生すると，クライアントでは平均 15 分 41 秒後に再開する．しかし，障害を発生させたファイルシステムノードにのみ保存され，複製が未作成のファイルに関しては，メタデータ上は存在するが実体ファイルが存在しない状態となる．障害復旧時に実体ファイルが残っていれば再度アクセス可能な場合がある．よって，自動複製ファイル作成機能により複製が作成されたファイルは消失を免れたため，ファイルシステムノードの各種障害に対する耐障害性はオープンソース版 Gfarm と比較して向上している．

ストレージシステム内に保存されている 50KB のファイル 20,000 個に対して複製の有無を確認するのに要した時間のグラフを図 6 に示す．このテストユーザーの複製数は 2 としている．運用中は自動複製ファイル作成機能により障害が発生しない限りはすべてのファイルが複製されていると想定する．この時，実体ファイル確認なしの場合は 17.5 秒，ありの場合は 169 秒掛かる．なお，実体ファイル確認を行うと処理時間が掛かるのは，メタデータで指定された各ファイルシステムノードに実体ファイルの存在確認を行うためである．また，再複製するファイルの数が増えると実体ファイル確認の有無による処理時間差が縮まるのはファイルの存在確認よりもファイルの再複製に時間が掛かるためである．この自己修復機能により，ファイルシステムノード障害発生時などに，短時間で必要なファイルだけを対象に再度複製を作成することが可能なため，ファイルシステムノード障害によるファイル消失を避けるための予防措置として有効であり，耐障害性が向上している．

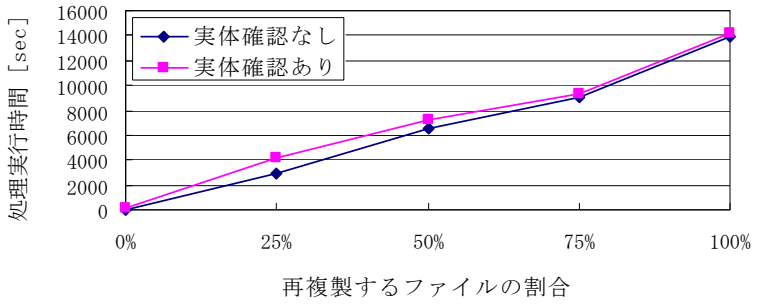

図 6 20,000 件のファイル複製確認に掛かる時間の推移

(3) ファイルの読み書き性能

本システムの性能評価を行うにあたり，本システム（複製あり，複製なし）とオープンソース版 Gfarm のそれぞれの環境で，事前検証としてファイルサイズを 16KB から 1GB まで，ブロックサイズを 512B から 1MB まで 2 倍ずつ変化させながら dd コマンドを利用して読み書きに要する時間を測定した．紙面の都合から一部のファイルサイズについてのみ結果を図 7 に示す．なお，図 7 は両対数グラフである．本システムで複製をしない場合との性能差は，読み込みは同等であるが，書き込みはブロックサイズが小さい場合に．40～70%の性能となる．他のファイルサイズの場合も図 7 と同様に，同一容量を読み書きする場合，ブロックサイズが 4KB 以上であれば性能は横ばいとなるため,ブロックサイズを意図的に小さくしなければ安定した結果が得られる．

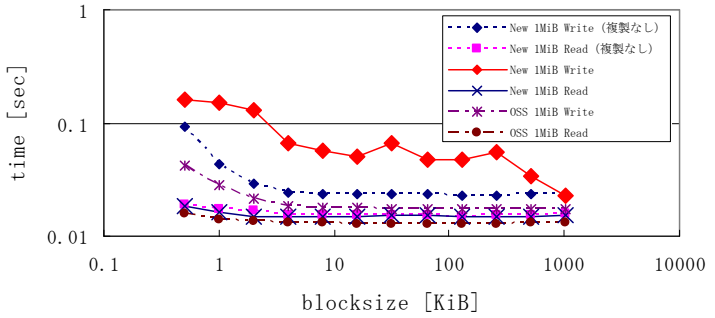

図 7 ブロックサイズの違いによる読み書き性能

ブロックサイズを事前検証で性能が良かった 1MiB に固定し，特定サイズのファイルを dd コマンドで読み書きに要する時間を測定した結果を図 8 に示す．なお，本システムではファイルの複製を行っているが，オープンソース版では複製は行っていない．この結果より，本システムはオープンソース版 Gfarm 比較と比較してファイルサイズに寄らず書き込みは約 80%，読み込みは約 90%の性能となる．性能差はファイルシステムノードの選択方法に差があるためであるが，ほぼ同等の性能である．

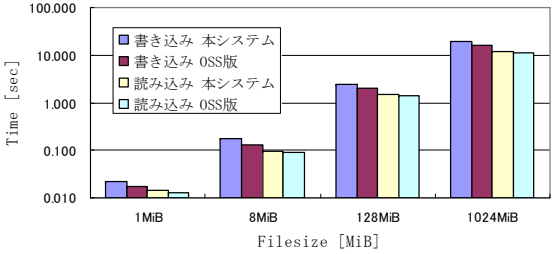

図 8 ファイル読み書き性能

4.2 アプリケーションと組み合わせた場合の評価結果

ストレージシステムは，アプリケーションサーバとの連携や，ファイル共有サーバとして動作させるなど何らかのアプリケーションと組み合わせて使われることが多い．そこで，大容量のストレージが必要とされるアプリケーションの一つとして市販されている電子メールのアーカイブソフトウェアと組み合わせてメールが消失しないか，性能が大幅に劣化しないかという点で評価を行う．市販パッケージとして DigitalArts 社製の m-FILTER[6]のクラウドバージョンを用いる．m-FILTER は，通常はサーバに直接接続されたハードディスクを利用するが，データベースソフトウェアのようなブロックアクセスではなくファイルアクセスであるため，fuse[7]を利用し，本システムをマウントしたクライアントサーバにインストールするだけで利用可能になるため選定した．

アプリケーションと組み合わせた場合の耐障害性評価方法は，1 ドメインに対して 50KB のテストメールを連続して送受信させている最中に本システム内で障害を発生させた場合に，テストメールの送受信が止まることがないか，テストメールを消失しないかどうかを確認する．各障害に対する詳細な影響については紙面の都合から記述しないが，一例としてメタデータサーバのネットワーク障害の場合を示す．メタデータサーバにネットワーク障害が起こると切り替わりが発生するため，テストメールの送受信中に行われる本システムに対しての読み書きに影響がある．すなわち，テストメールの送受信には影響はしないが，メールのアーカイブ処理が一時中断する．また，m-FILTER は内部動作として stat システムコールにより正常に書き込めたかどうかを確認する仕様であるため，書き込みに失敗したファイルについては再書き込みが実施される．そのためアーカイブされないメールは存在しない．

アプリケーションと組み合わせた場合の性能評価方法は，1 ドメインに対して 50KB のテストメールを連続して受信させている最中に保存されるテストメールの通数を 1 秒当たりに換算した値とする．従来の利用方法であるサーバの内蔵ストレージの場合と，本システムを組み合わせた場合の結果を表 2 に示す．本システムを利用した場合，1 秒あたりに保存される件数は 79%に低下するが，メールアーカイブにより増え続ける容量の限界は本システムの方が大きく，性能低下のデメリットよりも容量限界のメリットの方が大きい．

機能面，性能面から評価した結果，市販アプリケーションがファイルの書き込み完了後に確認を行う仕組みと，書き込みエラー発生時にリトライを実施する仕組みの 2 点を実装していることと読み書きの性能よりも容量を重要視する場合に，本システムと市販アプリケーションの組み合わせが有効になる．実際に本システムと m-FILTER を組み合わせてサービスを提供しており，検証期間も含めるとすでに 1 年以上安定して動作している．

表 2 アプリケーションと連携時の性能差

	サーバの内蔵ストレージ	本システム
処理件数 [通]	151.5	120.0
実現性のある最大容量	約 60TB (HDD:2TB×12 台(RAID6)×3)	約 4～60TB×ファイルシステムノード数

5. おわりに

オープンソース版 Gfarm2.4.0 の開発版をベースに企業向けサービスに耐えうる高品質かつ耐障害性を高める機能を新規開発した．オープンソース版 Gfarm で課題となっていたシステム内のシングルポイントをなくすだけでなく，障害発生時に起こり得る保存ファイル消失の危機を回避するために自律的なファイル複製作成とその維持管理を実現するとともに，障害発生時でもストレージサービスを継続できることを示すことで，オープンソース版 Gfarm よりも耐障害性が向上していること明らかにした．また，本システムは小容量から大容量のファイルまで幅広く利用可能なため，専用アプリケーションではなく市販アプリケーションと組み合わせて利用でき，商用サービスを提供できることを示した．

開発中は，ログメッセージの出力内容について障害発生箇所が特定できる方式をコミュニティに提案し，試験中に発生した問題は開発コミュニティに報告し，本システムだけではなくオープンソース版の品質向上にも寄与した．

謝辞 本稿の作成にあたりご協力頂いた皆様，ならびに本システムの開発にご協力頂いた皆様に，謹んで感謝の意を表する．

参考文献

1) Gfarm File System: http://datafarm.apgrid.org/
2) 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).
3) 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).
4) Lustre: http://wiki.lustre.org/index.php/Main_Page
5) The Apache Software Foundation: Cassandra, http://cassandra.apache.org/, (2009).
6) m-FILTER: http://www.daj.jp/bs/mf2/ .
7) FUSE Filesystem in Userspace: http://fuse.sourceforge.net/ .