Gfarm ファイルシステムによる広域ファイル共有

データインテンシブサイエンスを促進する基盤ソフトウェア

建部　修見

筑波大学計算科学研究センター

1　はじめに

Gfarm ファイルシステムは 2000 年より研究開発を続けているソフトウェアです。当時、CERN の LHC 実験計画がスタートし、LHC 加速器から生成される年間数十ペタバイトの実験データの解析、そのデータサイズを上回るシミュレーションデータの生成、という前代未聞ともいえる規模のデータ解析システムに関する議論が活発でした。これまでは、加速器を保持する計算センターでこれらのデータ解析は行われていましたが、LHC 実験規模のデータ解析となると CERN の計算センターだけではとてもまかないきれません。そのため、各国の計算センターで、協力してデータ解析をするということになったのです。そのようなとき、大規模データ処理を行うために、計算ノードのローカルディスクを利用し、性能をスケールアウトさせる、ファイル複製により複数の組織間で効率的にファイル共有を行う、という設計で Gfarm ファイルシステムの研究開発を始めました[1]。

2　Gfarm ファイルシステムの概要

Gfarm ファイルシステム[2]は、複数の組織間でのファイル共有、複数の組織による大規模データ解析のために設計されたファイルシステムです。各組織に設置されたストレージを単一の階層的名前空間（ディレクトリツリー）で束ね、単一のファイルシステムとしてアクセスすることが可能です。各組織からのアクセスは基本的には各組織のストレージに対し行われるように設計されており、広域環境においてもスケールアウトするアーキテクチャとなっています。各組織で自由にストレージの追加が可能で、ストレージの追加によりアクセス性能がスケールアウトします。遠隔の他組織に格納されているファイルも透明にアクセスできます。また、自拠点にファイル複製を作成してより高速にアクセスすることもできます。ファイル複製は、ファイルアクセス性能の向上だけではなく、ネットワーク、ストレージ障害時における耐故障性を向上させるために利用されます。

2.1 ソフトウェアコンポーネント

Gfarm ファイルシステムは、図 1 のようにファイルシステムメタデータを管理するメタデータサーバ（MDS）と、分散して設置されているストレージをアクセスするためのストレージサーバ（I/O サーバ）で構成されます。

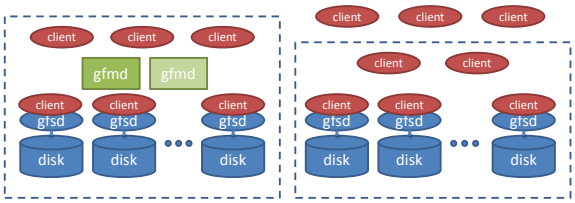

図 1 Gfarm ファイルシステムのソフトウェアコンポーネント

Gfarm ファイルシステムにおいて MDS は gfmd と呼ばれ、耐障害性を向上させるためマスタースレーブ構成をとることが可能です。I/O サーバは gfsd と呼ばれます。Gfarm ファイルシステムにおけるストレージは、ext3 などのローカルファイルシステムのディレクトリで、ディレクトリに対し I/O サーバを動作させることにより、そのディレクトリを Gfarm ファイルシステムの一部とすることが可能になります。

2.2 スケールアウトアーキテクチャ

MDS は、ディレクトリツリー、ファイル情報、ファイル複製格納位置などのファイルシステムメタデータを管理します。ファイルアクセスに際し、MDS はファイルのオープン時、クローズ時のみアクセスされ、実際の read、write、seek などのファイルアクセスは I/O サーバが直接アクセスされます。近くの I/O サーバ、またアクセス負荷の低い I/O サーバをそれぞれのクライアントがアクセスすることにより、アクセスを分散させることができます。これにより、MDS に対するアクセスがボトルネックとならない限り、全体としてクライアント数、I/O サーバ数に応じたスケールアウトするアクセス性能の実現が可能となります。

2.3 ソフトウェアリリース

Gfarm ファイルシステムのソフトウェア開発をオープンソースですすめるにあたり、Sourceforge.net を利用しています[3]。ここでは、最新リリースのダウンロード、不具合、要望などの登録、開発版、安定版ソースコードの閲覧、コミットログの確認などができます。

図 2 ダウンロード状況

図 2 は 2009 年 4 月 1 日から 2011 年 5 月 30 日までの約 2 年間のダウンロードの状況です。5,710 件のダウンロードがありました。

3 性能評価

本章では、スケールアウトするための条件となっている MDS の性能を示すとともに、クライアント数を増やしたときのファイルアクセス性能の評価を示します。

評価は科研費特定領域研究情報爆発 IT 基盤で構築された全国 14 拠点のクラスタからなる InTrigger プラットフォームを利用しました。評価にあたり Gfarm ファイルシステムの MDS は九州大に設置しました。I/O サーバは各拠点のクラスタの計算ノードで構成し、計 239 ノード、ストレージ容量は 146TB でした。クライアントは、I/O サーバにもなっている各拠点のクラスタの計算ノードです。各拠点から MDS までの RTT (往復遅延時間) は 0.04 ミリ秒～47 ミリ秒でした。Gfarm ファイルシステムはバージョン 2.3.1 を利用しました。

3.1 メタデータサーバの性能

MDS の性能評価にあたり、クライアント数を増やしたときのディレクトリの並列作成性能の評価を行いました。図 3 に評価結果を示します。X 軸は同時にディレクトリを生成するクライアント数、Y 軸は全体として一秒間に作成されたディレクトリ数です。クライアントは図に示される拠点に従い増加していきました。ディレクトリ作成は MDS への問合せが必要となり、クライアントからの一往復分のネットワーク遅延がかかります。そのため、少ないクライアント数では MDS の性能を飽和させることはできません。

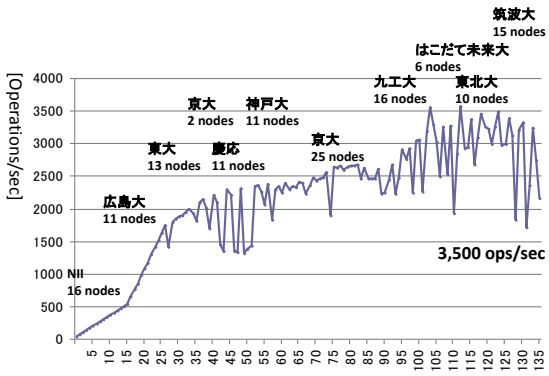

図 3 メタデータサーバ性能

占有していない広域ネットワークによる実測データであるため性能は安定していませんが、100 クライアントほどで毎秒 3,000 操作、最大で毎秒 3,568 操作の性能を示しました。これにより、MDS は毎秒 3,500 操作ほどの処理が可能であり、全体としての MDS アクセスがその性能を超えない限り MDS 性能はボトルネックとはならないことが分かりました。この性能までは、I/O 性能はスケールアウトすることが期待されます。

3.2 *N*ファイル並列書込・読込性能

次に、並列ファイルアクセス性能を示します。まず、図 4 にそれぞれのクライアントが 1GB の別ファイルを作成するときの性能を示します。MDS の性能評価と同様に、クライアントは図に示される拠点に従い増加させていきました。

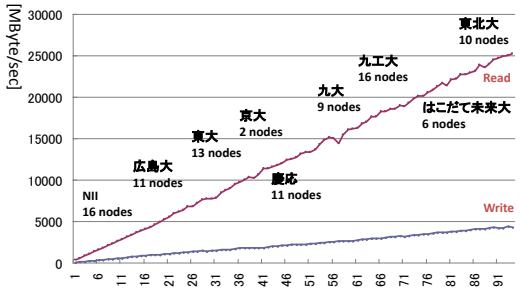

図 4 1GB の *N*ファイル書込・読込性能

各クライアントは I/O サーバでもあるため、各クライアントマシンのローカルディスクは Gfarm ファイルシステムの一部となっています。Gfarm ファイルシステムでは、ファイルの読込・書込に対し、ネットワーク的に近いストレージ、近いファイル複製を選択するようになっています。そのため、各クライアントが別ファイルを作成する場合、空き容量が十分あれば、各クライアントのローカルディスクが選択されます。並列書込については、それぞれのクライアントの書込はローカルディスクに対して行われ、書込 I/O に関するディスクアクセス競合はおこりません。MDS の性能評価で示されたように、MDS は毎秒 3,500 操作を行うことができます。そのため、94 クライアントからの並列アクセスでは MDS の性能はボトルネックとはなりません。結果として、I/O 性能はスケールアウトし、94 ノードで 4,370 MB/s の性能を達成しています。

並列読込は、並列書込で書き込んだファイルを読み込む性能です。書き込み時間は、書き込んだ後、ディスクに sync するまでの時間を計測していますが、読込は書き込んだ後すぐに行っているため、書き込んだデータがバッファキャッシュに載っており、性能が高くなっています。読込についても同様に MDS 性能はボトルネックとはならず、スケールアウトする性能を示し、94 ノードでは 25,170 MB/s の性能を達成しています。

3.3 1 ファイル並列読込性能

1 ファイルの並列読込性能を図 5 に示します。この評価では、1GB の 1 ファイルを各クライアントが並列に読み込みます。読み込むファイルは、あらかじめ各拠点に複製を作成しておきます。各拠点における複製数を 1、2、4、8 と変化させて性能を評価しました。これまでの評価と同様にクライアント数を図にかかれた拠点に従い増やしています。

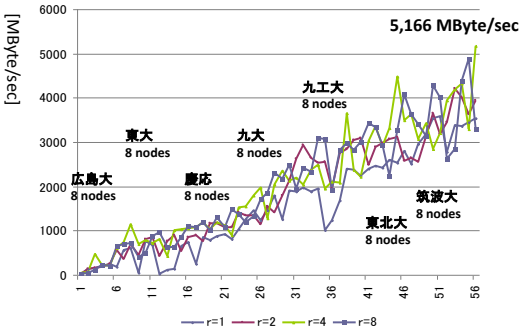

図 5 1GB の 1 ファイル読込性能

読込性能は安定していませんが、I/O 性能はスケールアウトし、7 拠点 56 クライアントで 5,166 MB/s の性能を達成しています。これにより、各拠点のクライアントは自拠点のファイル複製を参照していることが分かります。

4 基本的な利用方法

本章では Gfarm ファイルシステムの基本的な利用方法を説明します。

4.1 認証

Gfarm ファイルシステムをアクセスするために、まず認証の設定を行います。認証方法としては、共有鍵認証と GSI 認証をサポートしています。どの方式が利用可能かは管理者に問い合わせてください。

共有鍵認証の場合、gfkey コマンドで共有鍵を作成し、その鍵を MDS および全ての I/O サーバにコピーします。以下のように、31536000 秒 (約

1 年）有効の共有鍵を作成し、MDS など remotehost にコピーします。

```
% gfkey -f -p 31536000
% scp –p .gfarm_shared_key remotehost:
```

GSI 認証の場合は、以下のように代理証明書を作成します。

```
% grid-proxy-init
```

デフォルトでは 12 時間有効のものが作成されます。

4.2 マウント

Gfarm ファイルシステムは gfarm2fs コマンドによりマウントすることができます。gfarm2fs コマンドは利用者が書き込み可能なディレクトリに Gfarm ファイルシステムをマウントするコマンドです。たとえば、以下のようにマウントポイント/tmp/username を作成し、マウントします。

```
% mkdir /tmp/username
% gfarm2fs /tmp/username
```

アンマウントは fusermount –u を利用します。

```
% fusermount -u /tmp/username
```

Gfarm ファイルシステムは他の拠点でもマウントできますので、これで拠点をまたいでのファイル共有が可能となります。

4.3 Gfarm コマンド

通常のファイルシステムに対するコマンドは、gfarm2fs でマウントすることにより利用することが可能ですが、ファイル複製管理やクオータ機能など、Gfarm ファイルシステム独自の機能については Gfarm コマンドを利用します。

gfwhere コマンドはファイルの格納位置を調べるコマンドです。

```
% gfwhere .bashrc
linux-1.example.com
```

上記の例では、.bashrc は linux-1.example.com に格納されていることが分かります。

gfrep コマンドはファイル複製を作成するコマンドです。ファイル複製を二つ作成するためには以下のようにします。gfwhere コマンドでどこに作成されたか確認できます。

```
% gfrep -N 2 .bashrc
% gfwhere .bashrc
linux-1.example.com linux-2.example.com
```

gfrep コマンドは多くのコマンドオプションを持ち、複製作成先を指定したり、また必要数以上作成された複製を消去したりすることもできます。

そのほかのコマンドについては、Web ページ（http://datafarm.apgrid.org/software/gfarm_v2/html/ja/ref/）あるいはマニュアルページを参照してください。

5 最新機能・状況紹介

Gfarm ファイルシステムの新しい機能、特徴的な機能について紹介します。

5.1 ファイル自動複製機能

ファイルの作成時、更新時に自動的にファイル複製を作成する機能です。バージョン 2.4.1 以降でサポートしました。Google ファイルシステムをはじめファイル複製をサポートする多くのファイルシステムでは固定数のファイル複製しか作成できませんが、Gfarm ではファイル複製数はディレクトリの拡張属性 gfarm.ncopy で指定することができます。このため、ディレクトリによりファイル複製数を変えることが可能です。たとえば、/（ルート）ディレクトリ以下の全てのファイルに対し、複製数を 3 としたい場合は、以下のように/ディレクトリの gfarm.ncopy 拡張属性で 3 を指定します。

```
% echo –n 3 | gfxattr –s / gfarm.ncopy
```

このとき、ファイル作成時、更新時のファイルクローズ後に、非同期的にファイル複製を作成します。非同期に作成しますので、クライアントが複製作成完了まで待たされることはありません。

5.2 XML 拡張属性

バージョン 2.3.0 以降では、通常の拡張属性だけではなく値に XML 文書を持つ XML 拡張属性をもてるようになりました。XML 拡張属性は XPath で特定ディレクトリ以下のエントリを検索できるところに特徴があります。たとえば、/ディレクトリから foo というタグを含む XML 拡張属性を検索するためには以下のように行います。

```
% gffindxmlattr //foo /
```

/ディレクトリ以下の全てのエントリに対し、//foo という XPath 検索を行い、結果を表示します。

本機能により、アプリケーションのメタデータを、ファイルやディレクトリの XML 拡張属性に格納し、XPath 検索により該当ファイルを検索することが可能になります。

5.3 拡張 ACL 機能

バージョン 2.4.2 以降では、POSIX 1003.1e DRAFT 17 に基づく拡張 ACL をサポートしました。これにより、所有者、グループ、それ以外といった従来からの UNIX のファイルシステムのアクセス制御に加え、特定ユーザ、特定グループに対するアクセス制御が可能になります。

5.4 Gfarm ファイルシステムのフェデレーション機能

バージョン 2.4.2 以降では、Gfarm URL 形式（gfarm://metaserver:port/path/name）のシンボリックリンクをサポートしました。これにより、異なる Gfarm ファイルシステムのファイル、ディレクトリに対するリンクを作成することができます。利用者は、そのリンクを透明にたどることができますので、どの Gfarm ファイルシステムをアクセスしているのかを意識することなく、複数の Gfarm ファイルシステムをアクセスすることが可能です。

5.5 Debian パッケージ

Debian (Squeeze)のパッケージに Gfarm ファイルシステムが取り込まれました。

図 6 Debian パッケージ

これにより、Ubuntu を含む Debian 系のディストリビューションでは、apt-get install で Gfarm ファイルシステムをインストールすることが可能です。

6 大規模データ処理

Gfarm ファイルシステムは複数拠点間でのファイル共有だけではなく、スケールアウトするアーキテクチャにより、大規模データ処理用のファイルシステムとして利用することが可能です。ファイルアクセス性能をスケールアウトさせるためには、それぞれのクライアントが近くのストレージを、集中しないように分散してアクセスすることが大切です。大規模データ処理においては、特に入力ファイルの読込性能をスケールアウトさせるために、入力ファイルのファイル配置を考慮したジョブスケジューリングが重要となります。

以下、Gfarm ファイルシステムを利用した典型的な大規模データ処理について紹介していきます。

6.1 MPI-IO

アプリケーションから利用しやすいインターフェースに HDF5、Parallel netCDF などがありますが、それらは並列 I/O の標準インターフェースの MPI-IO を利用しています。

Gfarm ファイルシステムをマウントし、MPI-IO で Gfarm ファイルシステムを利用することも可能ですが、このとき MPI-IO では比較的典型的な N プロセスで 1 ファイルを並列に作成する場合にスケールアウトする性能が得られません。異なるファイル作成であれば、クライアントに近いストレージに分散して作成することによりスケールアウトする性能を達成することができますが、Gfarm ファイルシステムはファイルを分割しないため、1 ファイルへの並列書込は本当に 1 ファイルへの並列書込となってしまい、アクセスが集中してしまうためです。

そのため、そのようなケースもついても、意味を変えることなく、物理的には N ファイルに書き出しを行う MPI-IO の設計、実装を行っています[4]。性能評価によると、並列ファイルアクセス性能はスケールアウトし、4 ノード以上では PVFS

より良い性能を示しました。MPI-IO ライブラリのソフトウェアはまだ公開していませんが、今後公開を予定しています。

6.2 MapReduce

大規模データ処理では、MapReduce も利用されるようになりました。MapReduce のオープンソースの処理系として Hadoop があります。Hadoop では HDFS と呼ばれるファイルシステムが利用されます。ただし、HDFS は POSIX 準拠ではなく、たとえばファイルの修正もできません。そこで、HDFS ではなく Gfarm ファイルシステムを Hadoop MapReduce で利用する研究開発を行っています[5]。

Hadoop MapReduce も、マウントすることにより Gfarm ファイルシステムをそのまま利用することが可能ですが、それでは、マップタスクの配置に対し、入力データの配置を考慮することができません。そのため、不必要に遠隔アクセス、アクセス衝突が発生してしまいます。これを防ぐために、Gfarm ファイルシステムに対する Hadoop プラグインを開発し、公開しています[3]。

MapReduce の代表的なアプリケーションである、Grep（UNIX の grep とは違い、指定文字列の出現回数をカウントする）と Terasort の性能を図 7 に示します。

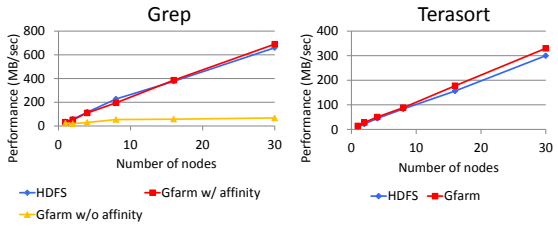

図 7 Grep と Terasort の性能

Grep では、青の HDFS と赤の Gfarm w/ affinity がほぼ同様のスケールアウトする性能を示しています。参考のため、Gfarm ファイルシステムを利用するものの、ファイル配置情報を利用しない場合の性能を、黄色の Gfarm w/o affinity で示します。ファイル配置が利用できない場合、マップタスクはデータの先頭から固定長ブロックごとを処理するように実行されますが、それらは物理的には単一ファイルに格納されているため、アクセスが集中してしまい、ノード数を増やしても性能向上していません。また、Terasort は、Gfarm ファイルシステムが HDFS を若干上回る性能を示しています。

プラグインの開発により、HDFS と同等あるいは HDFS をしのぐ性能を達成することができました。これにより、Hadoop MapReduce を使うために、HDFS を構築しデータを HDFS にコピーする必要はなくなり、全て Gfarm ファイルシステムにおいて、通常アクセス、MPI-IO による並列アクセス、MapReduce アプリケーションによる並列アクセスが可能となりました。

なお、Lustre、GPFS、PVFS などの並列ファイルシステムに対し Hadoop MapReduce アプリケーションを実行させるための研究もありますが、並列アクセス性能を向上させるため、ブロックサイズを通常より 1,000 倍も大きくするなど、通常とは大きく異なる設定が必要で、通常利用については逆に性能が落ちてしまいます。それに対し、Gfarm ファイルシステムは、もともと HDFS と設計が似ていることもあり、通常の設定のままで性能を出すことができます。

6.3 Pwrake によるワークフロー実行

大規模データ処理では、様々なプログラムを組み合わせてデータ処理を行うことが多く、それらをまとめてワークフローとして実行することも多く行われています。

ワークフローの記述はアイコンなどでグラフィカルに構成する Taverna、Kepler、スクリプト言語の Swift、MegaScript、依存関係を記述する Makefile などがあります。

Gfarm ファイルシステムに格納されているファイルに対し、効率的に大規模データ処理を行う場合、ファイルの格納位置を考慮したジョブのスケジューリングが重要となります。そのために、我々のグループでは Pwrake ワークフローエンジンの研究開発を進めています[6]。Pwrake は、UNIX におけるビルドツール make の ruby 版である rake を拡張し、同時に実行可能なジョブを並列分散実行します。ジョブの依存関係は rakefile により記述します。Rakefile は ruby で記述可能なため、極めて柔軟な記述ができます。複

雑な科学技術計算のためのワークフローでは、途中の実行結果によりワークフローが動的に変わるようなものも珍しくありませんが、そのような複雑なワークフローも記述可能です。

Pwrake では、並列分散実行の拡張だけではなく、Gfarm ファイルシステムにおけるファイルの格納位置を考慮したジョブスケジューリングを行います。

天文分野のデータ処理として、複数枚の天文画像を継ぎ合わせて一枚の天文画像にするモザイク処理があります。代表的なモザイク処理エンジンである Montage[8]によるケーススタディを紹介します。

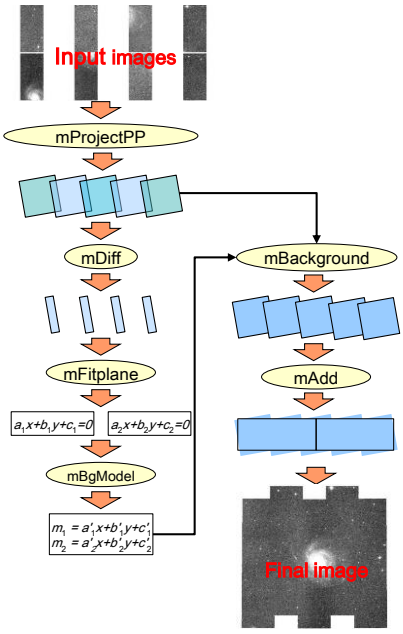

図 8 Montage ワークフロー

Montage のワークフローを図 8 に示します。複数の天文画像をまず同一平面上に射影し、オーバラップしているところで画像の明るさを揃えるなどの処理を行い、一枚のモザイク画像とします。天文画像は FITS 形式で保存されており、天体のどの場所をいつどのように撮影したものかなどのメタデータが画像ファイルのヘッダ部分に付加されています。ここで、オーバラップしている画像に対する処理をするために、全ての入力ファイルのヘッダ部分を解析して、どの画像がオーバラップしているかを調べる必要があります。つまり、実行開始時に全てのワークフローが決定しているわけではなく、ワークフロー実行中のジョブの出力に従って実行するワークフローが変わってきます。これまでのワークフローツールでは、このように動的にワークフローが決定されるものは扱えませんでしたが、Pwrake では rakefile により 100 行で完全に記述することが可能です。そのため、入力ファイルを指定するだけでワークフローの実行が可能となります。

Montage ワークフローに対し、クラスタのコア数を増やしながら実行時間をプロットしたものが図 9 です。

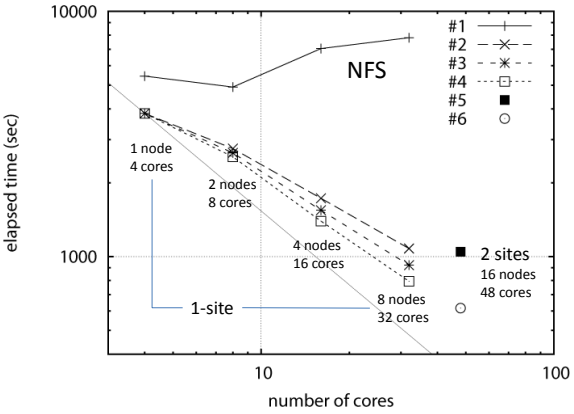

図 9 Montage ワークフローの実行時間

X 軸はワークフローを実行するコア数、Y 軸は実行時間です。データを NFS においた場合、コア数を増加しても実行時間が増えてしまっています。ディスクアクセス性能で実行時間が抑えられ、かつ並列アクセスによるアクセス衝突により全体性能が落ちています。Gfarm ファイルシステムでは、設定の違いにより三本の線が引かれていますが、どれもコア数を増やすと性能がスケールアウトしています。また、2 sites とかかれている筑波大と産総研の 2 拠点のクラスタによる広域分散データ処理では、初期データ配置を最適にすることにより■から○に性能向上し、1 拠点 32 コアから 2 拠点 48 コアに対し広域環境にもかかわらず性能がスケールアウトしています。

Pwrake および Montage ワークフローの rakefile は[7]で公開しています。

7 まとめ

Gfarm ファイルシステムは、2000 年から研究を進めているファイルシステムで、オープンソー

スで開発をすすめています。広域環境でも効率的にファイル共有が可能となるだけではなく、大規模データ処理に必要なスケールアウトするアクセス性能を達成できるようなアーキテクチャとなっています。MPI-IO、MapReduce、ワークフロー実行のそれぞれについて、効率的に処理するための研究開発も進めています。今後も引き続き研究開発を進め、あらゆる分野のデータインテンシブサイエンス（e-サイエンス）を促進することを目標としています。

参考文献

[1] 建部修見，森田洋平，松岡聡，関口智嗣，曽田哲之，“ペタバイトスケールデータインテンシブコンピューティングのための Grid Datafarm アーキテクチャ”，情報処理学会論文誌 : ハイパフォーマンスコンピューティングシステム，Vol.43，No.SIG 6 (HPS 5)，pp.184-195，2002

[2] Osamu Tatebe, Kohei Hiraga, Noriyuki Soda, "Gfarm Grid File System", New Generation Computing, Ohmsha, Ltd. and Springer, Vol.28, No.3, pp.257-275, 2010

[3] http://sourceforge.net/projects/gfarm/

[4] 木村浩希，建部修見，“広域分散ファイルシステム Gfarm の MPI-IO の実装”, 情報処理学会研究報告 2010-HPC-124(15)，pp.1-6，2010

[5] 三上俊輔，太田一樹，建部修見，”POSIX 準拠の広域分散ファイルシステム Gfarm 上での Hadoop MapReduce アプリケーション”，先進的計算基盤システムシンポジウム (SACSIS 2011) 論文集, pp.181-188, 2011

[6] Masahiro Tanaka, Osamu Tatebe, "Pwrake: A parallel and distributed flexible workflow management tool for wide-area data intensive computing", Proceedings of ACM International Symposium on High Performance Distributed Computing (HPDC), pp.356-359, 2010

[7] https://github.com/masa16/pwrake

[8] http://montage.ipac.caltech.edu/