Vol.2011-HPC-130 No.38
2011/7/28

Mogami: 高遅延環境において広帯域を達成する分散ファイルシステム

堀内美希[†1]　田浦健次朗[†1]

本研究の目的は，データ集約的な並列計算を行う際に重要な役割を担う，分散ファイルシステムの性能を，高遅延広帯域環境において向上させることである．現在の分散ファイルシステムは，高遅延環境で動作させると遅延の影響を受け，データ転送の際に広帯域を活かしきれていない場合がある．これを改善するため，本研究では積極的なデータプリフェッチ等により帯域を有効利用する分散ファイルシステム，Mogamiの提案・実装・評価を行い，高遅延環境において，広帯域が達成できていることを確認した．これにより，複数拠点にまたがる高遅延環境を含む分散環境でデータ解析を行う場合の性能向上が期待される．

Mogami: Distributed Filesystem that Utilizes Wide Bandwidth in High Latency Environment

Miki Horiuchi[†1] and Kenjiro Taura[†1]

This paper describes a distributed file system that utilizes wide bandwidth in high latency environments. Distributed file systems have been used as a way to share data for data intensive calculation. Existing distributed file systems, however, may fail to exploit the wide bandwidth due to high latency in the wide-area networks. To address this problem, we proposed and evaluated Mogami, a distributed file system that utilizes wide bandwidth even in high latency environment by aggressive data prefetching. In the evaluation we showed Mogami could achieve enough wide bandwidth to accelerate the data analyses in high-latency distributed environment.

†1 東京大学大学院 情報理工学系研究科

1. はじめに

大量の医学文書のインデクシングを行ったり，大量の天文学画像データを用いて超新星の発見を試みたりなどの，データ集約的計算を行うアプリケーションでは，複数の拠点間にまたがって大規模なデータ解析を行うことがある．その際，広域分散ファイルシステムによって，ローカルファイルを扱う感覚で透過的に共通のストレージを扱えると都合が良い．例えば，科学技術計算分野を例にとってみると，医学においては大量の医学文書を解析するMEDLINE to MEDIE[1)]，天文学においては大量の天文学画像データから超新星を発見することを目指したMontage[2)]等があり，その技術は広域分散ファイルシステムに基づいたワークフローに支えられている．

このように大規模なデータ解析技術を支えている広域分散ファイルシステムであるが，高遅延環境下でデータ転送を行うと実効帯域幅が物理的に可能とされる帯域幅と比べて大きく劣ることがある．これには様々な原因が考えられるが，真っ先に考えられることとして，分散ファイルシステムでは，データの転送要求と実際の転送をあるサイズごとに行うこととなるが，このサイズが小さい場合要求と転送を何度も繰り返すこととなり，特に高遅延環境では遅延の影響を受けるため広帯域が活かしきれない，ということがある．

今後，このようなデータ解析において扱うデータのサイズはますます増大し，さらにこのようなデータ解析が行われる環境が広域になってくるであろうことを考えると，広域分散ファイルシステムの高遅延広帯域環境への最適化は必須である．そこで本研究では既存の分散ファイルシステムにおいて無駄にされている物理的に可能な帯域を有効利用し，高遅延環境においても高い性能が出るような広域分散ファイルシステムMogamiを提案し，評価する．

2. 関連研究

2.1 分散ファイルシステム

広域環境でデータ集約的なアプリケーションを走らせる際に基盤として使用する分散ファイルシステムとして，Gfarm[3)]の研究，開発が行われている．Gfarmは，単一のメタデータサーバと複数のストレージサーバを束ねる構成をしており，Mogamiに近い構成である．ファイルシステム内部でファイル複製を管理し，耐故障性を実現したり，ネットワーク的に近いファイルにアクセスすることでネットワーク遅延を緩和する．また，多数のユーザ，グループの管理も可能としている．

Gfarm は，データ転送を行う際に現在の実装では 1MB 程の単位でファイルデータの要求とデータ転送を繰り返す．そのため，高遅延広帯域環境において大きなファイルを読み出す際には高遅延の影響を受けて広帯域を活かしきれないことがある．Gfarm を用いてデータ転送実験を行ってみたところ，物理的に可能とされる帯域幅が 10Gbps，iperf を用いたデータ転送時の帯域幅が 2.5Gbps，遅延が約 27msec の環境で，1GB のファイルを cat コマンドで読み出した結果，実効帯域幅は 130Mbps 程度となった．一度に転送するデータサイズを単純に大きくすれば，シーケンシャルな読み込み性能は改善されるが，ランダムアクセスの際には無駄なデータ転送が起こることとなってしまう．

他にも，Lustre[4]，Ceph[5]，GPFS[6] 等の分散ファイルシステムが存在するが，これらは広域環境で使用されることを想定して設計されていない．仮に広域環境で使用することが可能であったとしても，高遅延環境を想定した最適化は行っておらず，物理的に可能とされる帯域を満足する通信を行うことはできないと予測される．

高遅延環境における分散ファイルシステムでのデータ転送最適化に関しても研究が行われている．7) では，様々なアクセスパターンと回線遅延における遠隔ファイルアクセス性能を評価し，その評価に基づいて RPC バッファサイズの動的最適化を，人手を介さずに行う手法を示している．

2.2 分散環境でのデータ転送最適化

分散ファイルシステムに限らず，分散環境における並列アプリケーションのデータ転送最適化に関する研究は多数行われている．特に，MPI におけるプリフェッチやキャッシュに関する研究は，HPC 分野において重要である．MPI I/O における Parallel I/O を，ファイルのアクセスパターン分析に基づいたプリフェッチを行うことによって高速化する研究[8] や，GPFS を用いることを前提に，MPI I/O を GPFS のプリフェッチ機能を用いて高速化する研究[9] 等が行われ，MPI の高速化が図られている．

また，分散環境におけるデータ転送最適化といった観点からは，GridFTP[10] に関する研究も多数行われている．GridFTP は，遅延の大きい分散環境で大容量のデータを効率的に転送するためのデータ転送プロトコルで，複数の TCP コネクションを並列に確立することによりスループットの向上を図る並列データ転送の機能をサポートしている．この並列 TCP コネクション数をネットワーク環境に応じて適切に設定する機構も提案されている[11]．

2.3 分散環境でのワークフローの実行

分散ファイルシステムによって共有された（ように見える）共通のストレージが存在することを前提に，主に HPC 分野に焦点を当て，ワークフローを分散環境で簡単に実行するためのシステムの一つとして，GXP Make[12] がある．GXP Make は Makefile を記述し，その Makefile に記述された依存関係を元に，並列実行可能なジョブを判断し，分散環境で適切にジョブを並列実行するシステムである．

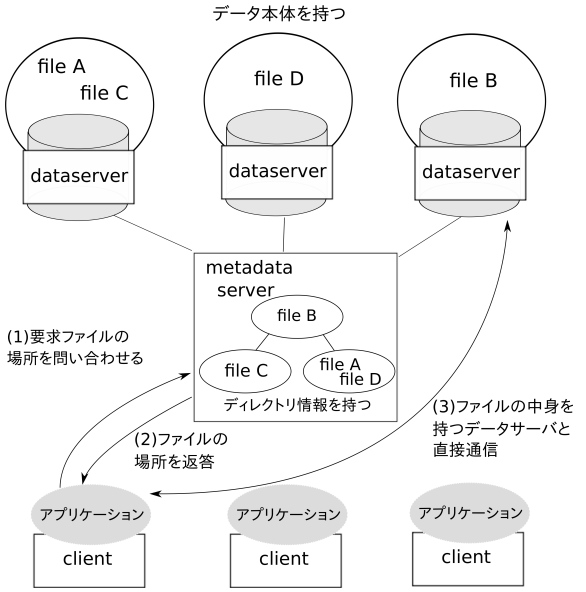

図 1 Mogami のシステムコンポーネント
Fig. 1 System components of Mogami

このようなワークフローアプリケーションの入力と出力は基本的にファイルであり，Mogami はそのデータの共有環境として使用されることを想定している．

3. 設　　計

3.1 システム構成

Mogami のシステムコンポーネントについて述べる．Mogami には，任意の数のクライアント・データサーバ，単一のメタデータサーバが存在する．（**図 1**）それぞれの役割について以下に記す．

クライアント（複数可）

Mogami ファイルシステムをマウントポイントを決めてマウントし，ファイルを使用す

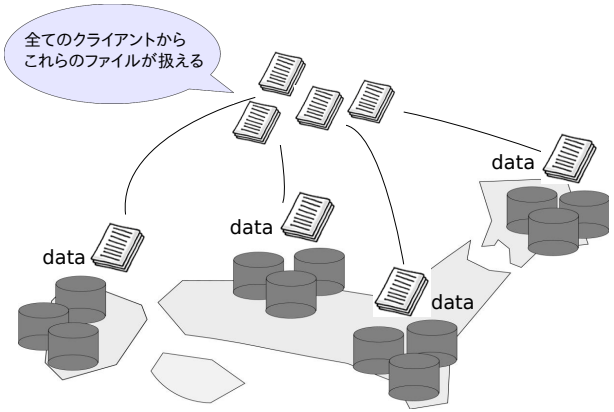

図 2 分散環境での使用イメージ
Fig. 2 Use in distributed environment

るクライアント．POSIX 準拠の API により，マウントポイント以下にアクセスすることによって実際のデータは遠隔ノードに存在するファイルをローカルに存在するかのように扱うことができる．

データサーバ（複数可）

ファイルの実データをファイルとして保存しており，クライアントからの読み込み要求に従って要求ファイルデータをクライアントに送信したり，逆に書き込み要求の際には通信で送られてきたデータに従ってファイルデータを書き換えたりする．

メタデータサーバ（単一）

各ファイルがどのデータサーバに保存されているかといった情報や，ディレクトリの木構造など，Mogami ファイルシステムにおけるメタデータ管理を行う．クライアントから要求があったときに，そのファイルをどのデータサーバが持っているかを通知する．また，メタデータの照会や変更要求があった際にはデータサーバを介さずに，クライアントとメタデータサーバのやり取りのみで処理が完結するようにする．Mogami ファイルシステム内で単一なノードである上に，全てのファイルアクセスはこのメタデータサーバと 1 回以上通信して行うことになるため，ここに極力負荷がかからないように注意したい．

以上 3 つのコンポーネントにより Mogami ファイルシステムは構成されているが，その

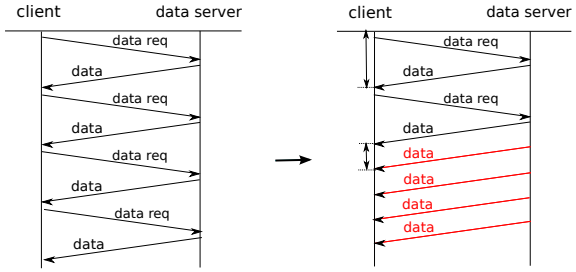

図 3 アプローチ：プリフェッチ
Fig. 3 Approach: Prefetching

働きを簡単に説明するために，図 1 中に Mogami でファイルの読み込み等のファイル操作をする際の手順を記す．まずクライアントがメタデータサーバに，目的ファイルを持っているデータサーバの位置を問い合わせ，返答されたデータサーバと一対一でデータ通信を行い，ファイルデータを操作する．

3.2 分散環境でのワークフロー実行システムとの組み合わせ

次に，GXP Make のようなワークフロー実行システムと組み合わせて使う分散ファイルシステムとしての設計を考える．分散環境でワークフローを実行するときには，その入力データはローカルのファイルシステムや NFS，Lustre 等のユーザが普段使用しているファイルシステム上に格納されているとする．その状態でワークフローを実行し，LAN 内に限らず複数拠点間でデータを共有する必要が生じるときに Mogami を使用する．ワークフローの各ジョブの入力，出力はファイルであるので，遠隔ノードに位置するそれらをワークフローを実行しているどのノードからも透過的にアクセスできるようにする役割を担う必要がある (**図 2**)．さらに，このときに起こるデータ転送の環境に合わせた最適化を自動で行うシステムとして，Mogami 分散ファイルシステムを設計する．

3.3 高遅延環境で広帯域を達成するためのアプローチ

広域分散ファイルシステム Mogami を設計・実装するにあたり，高遅延環境においても広帯域を達成可能とするためのアプローチとして，以下の 2 つのアプローチを考える．

- あるファイルをシーケンシャルに読んでいる際に，先のデータも使用されることを予測し，空いている帯域を効率的に用いてデータ転送を行っておく（プリフェッチ, **図 3**）
- TCP 輻輳制御パラメータの調節により，ACK が返ってくるまでに時間がかかる高遅延環境においても輻輳ウィンドウサイズを適切に調節し，物理帯域を有効利用できるよ

うにする

これらのアプローチは，高遅延環境で分散ファイルシステムを用いたデータ転送を行う際に広帯域であってもそれを使い切ることができない場合がある理由として，それぞれ以下を考えた場合に対する回避方法である．

- 高遅延環境で,（クライアントからデータサーバへの）データ転送の要求 → （データサーバからクライアントへの）データ転送を小さなサイズで繰り返し行うと，遅延の影響を受けて物理帯域が活かしきれない
- 高遅延環境で TCP 通信を行うと，遅延の影響を受けて ACK が返ってくるまでに（低遅延環境に比べて）時間がかかってしまい，輻輳ウィンドウサイズが適切に設定されないため物理帯域が活かしきれない

本研究では主に前者のアプローチ（プリフェッチ）に着目し，以下で実装と評価を行っている．

4. 実　　装

4.1 各コンポーネントの実装

4.1.1 クライアント

クライアントのファイルシステムの実装には，FUSE[13] を用いる．これにより，ユーザ空間のプログラム内でファイルアクセス API を定義し，ユーザ空間で動作する独自ファイルシステムを開発することができる．これを任意のマウントポイントにマウントすることにより，POSIX 互換の API でファイル操作が可能となるよう実装する．このクライアントのファイルシステム実装は Python で行い，ライブラリとして python-fuse を用いる．

Mogami では，クライアントは Mogami ファイルシステムをマウントした時点で指定されたメタデータサーバに接続を試み，アンマウントするまで保持される．データサーバへの接続は，ファイルを開くごとにファイルを持っているデータサーバに対して行い，ファイルを閉じるまでこのコネクションは保持する．また，後に実装については詳しく説明するが，プリフェッチを行うために，データサーバに対するコネクションをもう一つ作成しておく．このコネクションに関しても，ファイルを閉じたときに閉じる仕様とする．

クライアントには，上記の分散ファイルシステムとしての基本実装の他に，プリフェッチの実装を行う．この実装の詳細については 4.2 で詳しく述べる．

4.1.2 データサーバ

データサーバの実装に関して述べる．データサーバは，あるディレクトリを指定し，そのディレクトリ内に Mogami ファイルシステムで管理しているデータを保存する．ファイルを新規作成する際にどのデータサーバにデータが保存されるかは，現段階ではランダムで一台のデータサーバが選ばれる．機能としては，データサーバでは常にクライアントからの接続待ちをしており，クライアントから接続要求があるとスレッドを新規作成しそのスレッドに接続要求のあったクライアントに対するデータ送受信の処理を任せる．

主にデータサーバがクライアントから受け取る要求は，ファイルデータの読み書き要求であり，読み込み要求もプリフェッチ要求も，データサーバ側から見れば同じく要求ファイルの，指定された位置のデータを送信する処理である．

4.1.3 メタデータサーバ

次に，メタデータサーバの実装に関して述べる．サーバとしての基本的な仕組みはデータサーバと同じであり，複数クライアントからの接続要求を受け付けることが可能な実装を行う．メタデータサーバはクライアントと常に TCP コネクションを保持しており，ファイルのデータ読み書き以外の操作では，常にクライアントとデータの送受信をする．メタデータサーバに関しても，あるディレクトリを指定して起動し，そのディレクトリ以下に Mogami ファイルシステムのディレクトリ構造と同じディレクトリ構造を作成していく．つまり，新規ファイルやディレクトリの作成要求があった場合に，実際にメタデータサーバの指定したディレクトリ以下にファイルやディレクトリを作成する．ただし，データ本体はデータサーバが持ち，メタデータサーバには存在しない．その代わりにメタデータサーバで実際に作成したファイルの中には，どのデータサーバがこのファイルデータを持っているのか，という情報を書き込んでおき，適宜参照しクライアントに通知する．その他にも，ファイルのタイムスタンプ，所有者情報やパーミッション等の情報を持っており，要求に応じてクライアントに通知する役割がある．

4.2 プリフェッチ

Mogami では，高遅延環境でも広帯域を達成するためにプリフェッチを用いるが，以下にその実装について述べる．プリフェッチは，あるファイルをシーケンシャルに読み込んでいる（と Mogami が判断した）際に行うこととする．つまり，前回の読み出しでアクセスした最後の位置を記憶しておき，その次の読み出しが前回の続きの位置からアクセスした場合に，クライアントからデータサーバにプリフェッチの要求が出されることとなる．

プリフェッチを行うときの条件について述べたが，次に，プリフェッチが起こる場合のリクエストの出し方等の流れについて述べる．**図 4** にプリフェッチが起こる場合の，プリフェッチの流れを図示している．まず，Mogami ではファイルを開いたときに，クライアントでプ

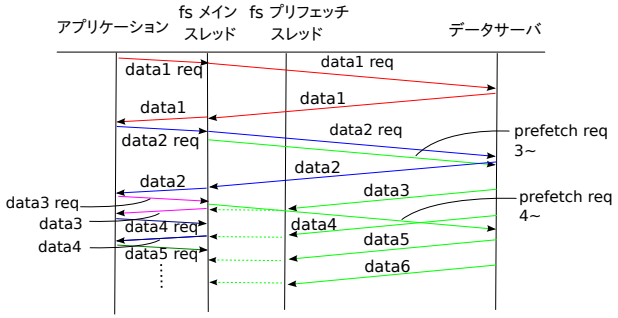

図 4 プリフェッチの流れ
Fig. 4 Flow of prefetching

リフェッチ用のスレッドが立てられる．(図 4 中の fs プリフェッチスレッド) そして，ファイルの読み出し要求がシーケンシャルに行われた場合，fs メインスレッドからデータサーバにプリフェッチの要求が行われる．その後，プリフェッチ要求を受けとったデータサーバは，受け取った要求に合わせてファイルのデータを fs メインスレッドではなく fs プリフェッチスレッドに転送する．fs プリフェッチスレッドはこれをクライアントの共通のバッファに格納することで，データをプリフェッチしておくことが可能となる．

ここで，fs メインスレッドでデータを読み込む際に，まずデータ読み込みの際には，要求位置のデータがバッファに既にプリフェッチされていないか調べ，されている場合はそのデータを用いる．また，まだバッファに格納はされていないが，データサーバにプリフェッチ要求を行っているという場合には少し時間をおいてからバッファを再びチェックする，という実装が必要となる．

このプリフェッチで要求するサイズであるが，現在は設定ファイルに直接数値を書き込むことになっている．5.2 で，このサイズを変更したときのファイルのシーケンシャル読み込み性能評価を示す．

5. 性 能 評 価

5.1 実 験 環 境

以降で本研究で提案する Mogami に関しての性能評価を行うが，まずは実験環境について述べる．実験環境としては InTrigger プラットフォーム[14] を使用する．広帯域高遅延環境である，tsukuba（つくば）-huscs（札幌）クラスタのマシンを用いた．それぞれのマシンの基本スペックは**表 1** に，各ネットワーク環境は**表 2** に示す．

表 1 実験マシンスペック
Table 1 Machine configurations used in evaluations

ノード	CPU	メモリ	OS
tsukuba000-014	Intel Xeon E5410 8cores	32GB	Linux2.6.26 (64bit)
huscs000-019	Intel Xeon E5530 8cores(w/ HT 16cores)	24GB	

表 2 実験ネットワーク環境
Table 2 Network configrations used in evaluations

	物理帯域	RTT
tsukuba-huscs 間	1Gbps	約 27msec

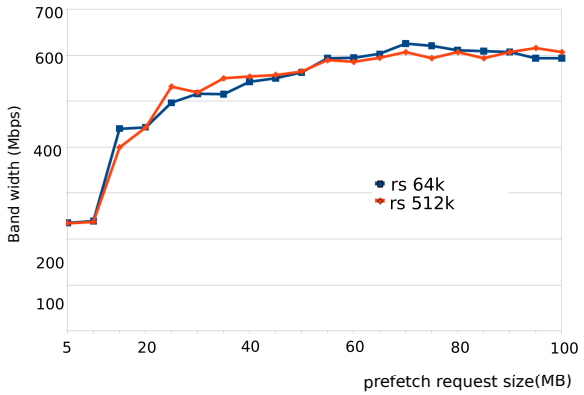

図 5 プリフェッチ要求サイズの変化によるファイル読み出し性能評価
Fig. 5 Performance of file reading by prefetching size

5.2 プリフェッチ要求サイズとファイル読み出し性能の関係

Mogami において，プリフェッチ要求が行われる際にどれくらい先までプリフェッチ要求を出すかというパラメータを変化させた際の，ファイル読み出し性能の評価を行う．ブロックサイズは 1MB，読むファイルサイズは 1GB に固定して実験する．

用いたプログラムは，指定したファイルを開き先頭からシーケンシャルにある読み込みサ

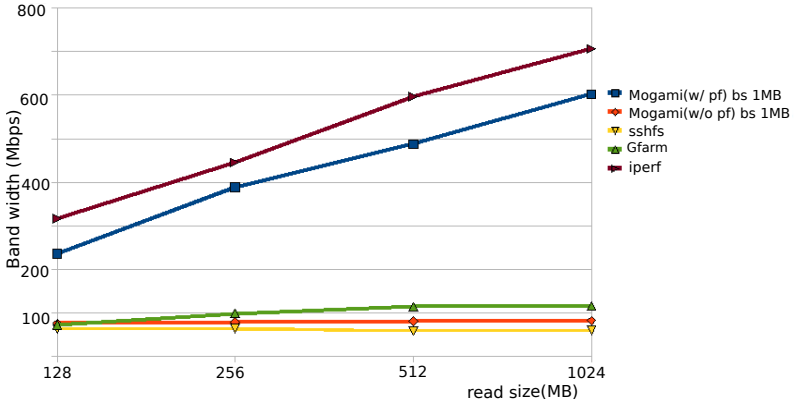

図 6 シーケンシャル読み出し性能比較
Fig. 6 Comparison of file reading performance

イズ（ここでは 64kB と 512kB の 2 種類，グラフ中では rs と記述する）ずつ読むことを繰り返して，ファイル末尾に到達するまでの時間を測定するプログラムである．

測定を行った結果を**図 5** に示す．これらの結果から，プリフェッチ要求を出す際にどれくらい先まで要求を出せば最も効果的かというと，tsukuba-huscs 間では約 70 ～ 80MB であると言える．これよりも少ない数のプリフェッチ要求しか出さなかった場合，ファイルを読み出すスピードがプリフェッチに追いついてしまい，性能は低下する．逆にこれより多い場合には，プリフェッチ要求を出すオーバヘッドが大きくなってしまい，性能は向上しない．

5.3 ファイルのシーケンシャル読み出し性能評価

高遅延環境下での使用に最適化した広域分散ファイルシステム Mogami の性能評価を行う．Mogami は高遅延環境下でのデータ読み込み実効帯域幅の向上を目標としている．そこで実際に高遅延広帯域な実験環境において，ファイルのデータ読み込みを行い，実効帯域幅を測定することによって評価を行う．tsukuba-huscs 拠点を用い，tsukuba に存在するファイルを huscs で読み込んだ場合の，データ転送帯域幅を iperf，Gfarm，sshfs[15]，Mogami 間で比較する．ファイルの読み込みは，一回につき 64KB ずつ，ファイルの先頭から末尾までシーケンシャルに行う．

性能評価として，読み込んだファイルの大きさと実効帯域幅の関係を，**図 6** に示す．Mogami でプリフェッチを行ってデータを先読みした時の実効帯域幅は，1GB のファイルを読み込んだ時に関して述べると，Mogami でプリフェッチを行わない時や他ファイルシステムの 6 ～ 10 倍程になっている．結果として，プリフェッチを用いると，iperf には少し及ばないものの，ファイル読み込みの帯域幅をかなり大きくすることができることを確認できた．

また，今回は評価実験ではランダムにファイルを読み込んだ場合について言及していないが，Mogami の場合，シーケンシャルにファイルが読み込まれている時のみプリフェッチを行うため，ランダム読み込み時に無駄なデータ転送を行うことはない．

6. おわりに

6.1 まとめ

本研究では，データ集約的な並列計算を行う際に重要な役割を担う分散ファイルシステムの中でも，高遅延環境においても広帯域を達成するような分散ファイルシステムとして，Mogami の提案，設計，実装を行った．結果として Mogami を用いたデータ転送では，高遅延環境でも物理的に可能とされる帯域に近い実効帯域を達成できることを実測し確認した．

6.2 今後の課題

本研究の今後の課題として，以下の方針を考えている．

- プリフェッチ要求を出すサイズの環境に応じた自動チューニング
- GXP Make 等のワークフロー実行システムと簡単に組み合わせて使うことができる分散ファイルシステム
- 広域環境でワークフローアプリケーションを走らせた場合の複数クライアントの通信競合回避
- ワークフローアプリケーション実行時に RAM Disk 等の資源を適切に活用

参考文献

1) Chun, Hong-woo, Tsuruoka, Y., Kim, J.-D., Shiba, R., Nagata, N., Hishiki, T. and Tsujii, J.: Extraction of Gene-Disease Relations from MedLine using Domain Dictionaries and Machine Learning, *The Pacific Symposium on Biocomputing (PSB) Maui, Hawaii, USA*, pp.4–15 (2006).
2) Montage : An astronomical image mosaic engine. http://montage.ipac.caltech.edu/.
3) Osamu Tatebe, K. H. and Soda, N.: Gfarm Grid File System, *New Generation Computing*, Vol.28 (2010).
4) Lustre file system :. http://www.lustre.org/.
5) Weil, S. A., Brandt, S. A., Miller, E. L., Long, D. D. E. and Maltzahn, C.: Ceph: A scalable, high-performance distributed file system, *Proceedings of the 7th symposium on Operating systems design and implementation*, OSDI '06, Berkeley, CA,

USA, USENIX Association, pp.307–320 (2006).
6) Schmuck, F. and Haskin, R.: GPFS: A shared-disk file system for large computing clusters, *Proceeding of the Conference on File and Storage Technologies*, pp. 231–244 (2002).
7) 大辻弘貴, 建部修見: アクセスパターンと回線遅延を考慮した遠隔ファイルアクセスの最適化, *SACSIS*, pp.11–19, (2011)
8) Byna, S., Chen, Y., Sun, X.-H., Thakur, R. and Gropp, W.: Parallel I/O prefetching using MPI file caching and I/O signatures, *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, SC '08, Piscataway, NJ, USA, IEEE Press, pp.44:1–44:12 (2008).
9) Prost, J.-P., Treumann, R., Hedges, R., Jia, B. and Koniges, A.: MPI-IO/GPFS, an Optimized Implementation of MPI-IO on Top of GPFS, *SC Conference*, Vol.0, p.58 (2001).
10) Allcock, W., Bester, J., Bresnahan, J., Chervenak, A., Liming, L., Tuecke, S., Memo, S. O.T., Liming, L. and Tuecke, S.: GridFTP: Protocol extensions to FTP for the Grid, *GWD-R (Recommendation*, p.3 (2001).
11) 伊藤建志, 大崎博之, 今瀬眞: GridFTP-APT : データ転送プロトコル GridFTP の並列 TCP コネクション数調整機構 (インターネットの測定・性能評価技術及び一般), 電子情報通信学会技術研究報告. IN, 情報ネットワーク, Vol.105, No.472, pp.19–24 (2005-12-08).
12) Taura, K., Matsuzaki, T., Miwa, M., Kamoshida, Y., Yokoyama, D., Dun, N., Shibata, T., Jun, C.S. and Tsujii, J.: Design and Implementation of GXP Make – A Workflow System Based on Make, *eScience, IEEE International Conference on*, Vol.0, pp.214–221 (2010).
13) FUSE :. http://fuse.sourceforge.net/.
14) InTrigger Platform :. http://www.intrigger.jp/.
15) sshfs :. http://fuse.sourceforge.net/sshfs.html.