FUSE の高速化手法を適用した Gfarm の並列アクセス性能

石黒 駿[†1] 村上 じゅん[†1] 大山 恵弘[†1,†2]

1. はじめに

FUSE[1] は，ユーザレベルファイルシステムを実装するためのフレームワークである．FUSE を利用して実装されたユーザレベルのファイルシステムは，カーネルモジュールとデーモンで構成される．FUSE の動作を図 **1** に示す．

アプリケーションのプロセスがユーザレベルファイルシステムへ発行した read や write などのシステムコールは，カーネルモジュールによって受理される．次にカーネルモジュールは，そのシステムコールの実際の処理をデーモンに依頼する．デーモンは，その要求にしたがってローカルファイルシステムからファイルを読み込んだり，リモートのファイルを読み込むなどの実際の処理を行い，その結果をカーネルモジュールへ返す．カーネルモジュールはその結果を受け取り，システムコールの処理が完了する．

しかしながら，デーモンがローカルファイルシステムのファイルをそのまま読み書きする場合，プロセスが直接そのファイルを読み書きする場合と比較して処理が冗長となる．例えば，プロセスが直接ローカルファイルシステムのファイルを読み込むときは，ローカルファイルシステムのキャッシュからプロセスのバッファへの 1 回のメモリコピーで済む．一方 FUSE を利用する場合，ローカルファイルシステムからデーモンのバッファへのメモリコピー，デーモンのバッファからカーネルモジュールのバッファへのメモリコピー，カーネルモジュールのバッファからプロセスのバッファへのメモリコピーと，3 回のメモリコピーを通して読み込み処理が完了する．

この問題に対して，石黒らの研究[2] がある．この研究では，カーネルモジュールからローカルファイルシステムに直接アクセスすることにより，ファイル読み書きの際のメモリコピーの回数を削減している．さらにファイル読み書きの際にデーモンを経由しないため，プロセスとデーモン間のコンテキストスイッチが発生しなくなる．

†1 電気通信大学
The University of Electro-Communications
†2 独立行政法人科学技術振興機構，CREST
JST，CREST

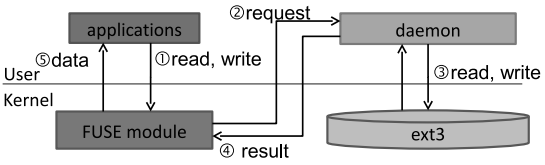

図 1 FUSE のアーキテクチャ

本研究では，この手法を Gfarm に適用し，複数のプロセスから Gfarm ファイルシステムに並列アクセスした場合の性能を評価した．その結果，適用前と比べて適用後は読み書き性能が向上した．

2. Gfarm

Gfarm[3] は広域ネットワークで利用可能な分散ファイルシステムである．Gfarm はメタデータサーバ及び I/O サーバから構成される．メタデータサーバは，Gfarm ファイルシステム上のファイルのメタデータやファイルの位置情報などを一括して管理する．I/O サーバは複数存在し，ファイルの内容を保存する．Gfarm のクライアントが Gfarm ファイルシステム上のファイルにアクセスする際は，まずメタデータサーバへアクセスし，どの I/O サーバに目的のファイルがあるかを知り，次に I/O サーバにアクセスしてデータをやりとりする．I/O サーバは Gfarm のクライアントと同じノードに配置可能である．Gfarm の特徴として，ローカルの I/O サーバを優先することで，高速にファイル読み書きができるという点が挙げられる．

Gfarm のクライアントは，FUSE を利用して，Gfarm ファイルシステムをマウントできる．通常クライアントは Gfarm ファイルシステムにアクセスするために，専用の API を用いる必要があるが，マウントすることにより，システムコールを利用して Gfarm ファイルシステムにアクセスできる．

3. 高速化手法

Gfarm を FUSE を用いてマウントする場合のデーモンは，gfarm2fs である．高速化手法では，gfarm2fs がローカルファイルシステムのファイルを開くと，以降は gfarm2fs を経由せずに，そのファイルを読み書きできる．その様子を図 **2** と図 **3** に示す．

ⓒ 2012 Information Processing Society of Japan

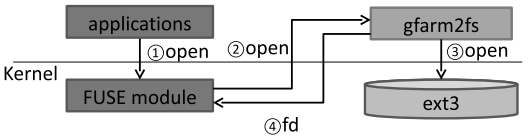

図 2　open 時の処理

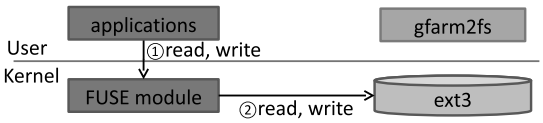

図 3　read と write 時にローカルファイルシステムに直接アクセスする様子

高速化手法は，open システムコールが発行されたときおよび，read システムコールと write システムコールが発行されたときに動作する．まず，open システムコールが発行されると，通常どおり gfarm2fs に open リクエストが要求される．gfarm2fs は目的のファイルが配置された I/O サーバの位置をメタデータサーバから取得し，そのファイルがローカルの I/O サーバに存在する場合は，gfarm2fs が直接そのファイルを open する．そしてそのファイルのファイルディスクリプタを open リクエストの結果とともにカーネルモジュールへ返す．カーネルモジュールはそのファイルディスクリプタを元に，そのファイルを特定する．以降はそのファイルに対する read システムコールと write システムコールが発行されたときは，カーネルモジュールは gfarm2fs にリクエストを送らず，直接そのファイルを読み書きする．

4. 評　　価

IOR HPC Benchmark を用いて，マウントされた Gfarm ファイルシステムに対して複数プロセスから並列にファイルアクセスを行った場合の性能を測定した．メタデータサーバ 1 台，クライアント兼 I/O サーバのノード 2 台の計 3 ノードであり，クライアントのノードに Gfarm ファイルシステムをマウントした．このうち，クライアントのノード 2 台で合計 24 プロセスを実行し，Gfarm ファイルシステムの並列アクセス性能を測定した．各ノードでそれぞれ 12 プロセスを動作させ，各プロセスがそれぞれ 8GB のファイルを 1MB 単位で読み書きするように設定した．

実験環境は，すべてのノードの性能が，CPU が Intel Xeon 2.40GHz × 2，メモリが 48GB，HDD が 15,000rpm 600GB であり，ノード間は InfiniBand にて接続されている．また OS は Cent OS 5.5 64bit（kernel-2.6.18），Gfarm のバージョンは 2.4.2，FUSE のバージョンは 2.7.4，Gfarm2fs のバージョンは 1.2.3 である．実験の結果を図 **4** に示す．

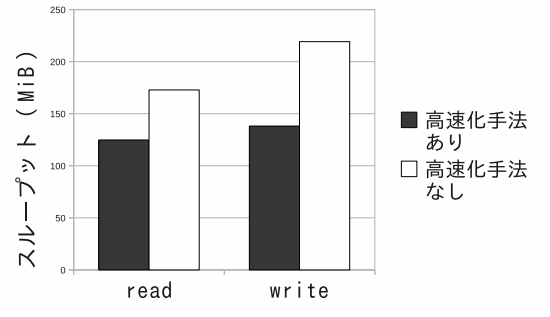

図 4　高速化手法適用前と適用後の read と write のスループット

実験の結果，read 性能が高速化手法適用前と比べて高速化手法適用後で約 1.4 倍，write 性能が 1.6 倍に向上した．並列アクセスの場合でも，逐次アクセスと同等の性能向上が確認できた．Gfarm ではローカル I/O を優先するため，高速化手法の効果は高いと考えられる．

5. 現状と今後

FUSE を利用してローカルファイルシステムを読み書きする場合に，デーモンを経由せず直接ローカルファイルシステムを読み書きする手法を分散ファイルシステム Gfarm に適用し，複数プロセスから並列アクセスした場合の性能を測定した．その結果，read の場合で 1.4 倍，write の場合で 1.6 倍に性能が向上した．今後は，実際のアプリケーションを用いて性能評価を行う予定である．

謝　　辞

本研究を行うにあたって，有益な助言を頂いた筑波大学建部研究室の方々に深く感謝する．また本研究は，科学技術振興機構戦略的創造研究推進事業（JST CREST）の研究課題「ポストペタスケールデータインテンシブサイエンスのためのシステムソフトウェア」の支援を受けている．

参　考　文　献

1) Szeredi, M.: FUSE: Filesystem in Userspace. `http://fuse.sourceforge.net/`.
2) 石黒駿，村上じゅん，大山恵弘：広域分散ファイルシステム Gfarm におけるローカルストレージアクセスの高速化，ハイパフォーマンスコンピューティングとアーキテクチャの評価に関する北海道ワークショップ（HOKKE-19）(2011).
3) Tatebe, O., Hiraga, K. and Soda, N.: Gfarm Grid File System, *New General Computing*, Vol.28, No.3, pp.257–275 (2010).

ⓒ 2012 Information Processing Society of Japan