情報処理学会論文誌　コンピューティングシステム　Vol. 4　No. 4　122–134 (Oct. 2011)

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

大辻弘貴[†1,†2]　建部修見[†1,†2]

広域分散ファイルシステムにおける遠隔ファイルアクセスの性能は，回線遅延やアクセスパターンに大きな影響を受ける．本論文では様々な条件下で遠隔ファイルアクセスの性能を評価し，システムの設定が性能に与える影響を調査した．そして，その結果に基づき自動的に設定を最適化する手法を考案した．この手法により，既存の遠隔ファイルアクセス方式では大幅に性能が低下する高遅延環境や非シーケンシャルのアクセスにおいて，大幅な性能向上を達成した．あわせて，本提案手法による性能向上が有意なものであることを検証し，実測最大値に近い性能を示すことを確認した．

Optimization of Remote File Access Considering Access Pattern and Network Delay

Hiroki Ohtsuji[†1,†2] and Osamu Tatebe[†1,†2]

Performance of remote file access in distributed file system depends on network latency and access pattern. In this paper, we investigated effects to the performance of system configurations by evaluating performance of remote file access under various conditions. Then, we invented a method to optimize configuration automatically based on the result of the investigation. This method improves the performance of remote file access significantly under the high network delay condition where the existing method demonstrates lower performance. We also validate whether the performance improvement by the proposed method is significant and affirm that the improved performance is nearly maximum measured performance.

†1 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba
†2 独立行政法人科学技術振興機構，CREST
Japan Science and Technology Agency, CREST

1. 序　　論

増加の一途をたどるデータを取り扱うため，広域環境でデータを共有する必要性が高まっている．特に e-サイエンスやデータインテンシブコンピューティングの発展により，複数のスパコン間における大規模データ共有や，地理的に離れた複数拠点共同での解析する機会も増えている．そのような背景で，広域分散ファイルシステムが広く用いられている．広域分散ファイルシステムではデータ共有の高速化のためにファイル複製[1)]が行われることがあるが，クライアントにストレージが十分にない場合や，ファイルの一部分のみが必要な場合には遠隔ファイルアクセスを行う必要がある．ファイル複製時のバルク転送と異なり，遠隔ファイルアクセスは回線遅延の影響を大きく受ける．たとえば，高遅延環境におけるランダムアクセスは，LAN 内などの低遅延環境の場合と比較して大幅に性能が低下する．**図 1** は既存の方式における性能評価であり，低遅延環境では十分な性能を示していても，ランダムアクセスや高遅延環境での動作では性能が低下することが分かる．このような性能低下に対処する場合，管理者が手動で設定を最適化することがあるが，考慮すべき要素が多いうえ，状況の変化に対応することは多大な労力を要する．そこで，本論文では，分散ファイルシステムにおいて単一クライアントがネットワーク経由で単一のサーバに対してファイルアクセスを行う部分にフォーカスし，アクセスパターンや回線遅延・回線帯域を考慮して遠隔ファイルアクセスを自動的に最適化する手法およびその評価を示す．

2. 関 連 研 究

2.1　遠隔ファイルアクセスの方式

遠隔ファイルアクセスには大きく分けて 2 つの方式がある．1 つは遠隔手続き呼び出し（RPC）によってファイルのオフセットとサイズを指定して必要な部分を転送する方式と，Web のようにファイル全体を転送する方式である．前者には Gfarm[2)] ファイルシステムや NFS[3)]，PVFS[4)]，Lustre[5)] がある．これらの方式では，サーバとクライアントの間で，あらかじめ設定した一定量のデータを 1 つの単位としてやりとりを行う．このサイズは Gfarm バージョン 2.4.1 においては 1 MB の固定値となっており，NFS ではバージョンによって異なるが数十 KB である．後者の方式には AFS[6)] やその後継である Coda[7)]，FTP の拡張である GridFTP[8)] があげられる．

本研究が対象とするのは前者の RPC による転送を行う形式である．

2.2　遠隔ファイルアクセスの性能向上例

遠隔ファイルアクセスの性能向上の例としては，PVFS がファイルを分割して複数の I/O サーバに保管してストライピングを可能としている．また，Lustre[5] も同様にストライピングを行ったり，同一ファイルに対する複数プロセスからのアクセスを集約したりと高速化が図られている．また，サーバとクライアント間では，回線遅延のある環境を考慮して複数の転送命令を同時に発行することもできる．ただし，この場合にも転送データ量の単位など，固定されたパラメータが存在しチューニングが必要となる．そのほかにも，FTP をグリッド環境向きに拡張して作られた Grid FTP があり，複数の TCP コネクションを同時に利用することで性能を向上させる機能が備わっている．この TCP セッションの数は基本的に固定値であるが，状況に合わせて最適化する研究も行われている．文献 9) は実際に計測されたスループットをもとにその時点で最適な TCP セッション数を探索する手法で，実環境に近い条件下においても高い性能を示している．

文献 10) は Deduplication と呼ばれる，ファイルの重複部分の検出と集約によって，ネットワーク帯域およびストレージ領域を節約する手法について，利用の有無を状況に応じて動的に変更する研究である．本研究とは最適化する対象は異なるが，状態をモニタし，それによってアクションを起こすという点では共通している．

NFS もまた，Compound RPC という最適化手法を持っており，これはある程度のリクエストをまとめて 1 つの RPC に入れることで，通信回数を削減して回線遅延の影響を低減するものである．本研究は次章で述べるように，同期型 RPC と呼ばれる方式を対象としており，このアプローチとは前提条件が異なる．

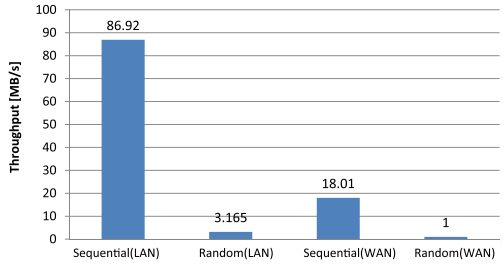


図 1　既存システム[2] における性能．LAN のネットワーク遅延は 50 μs，WAN は 25 ms
Fig. 1　Performance of existing method. Network delay of LAN is 50 μs and WAN is 25 ms.

2.3　アクセスパターンの解析例

本研究では，アクセスパターンの認識も行っている．文献 11) は，遠隔ファイルアクセスではないがアクセスパターンをニューラルネットワークによって分類し，キャッシュなどのポリシをアダプティブに適用した研究である．

3.　遠隔ファイルアクセスの方式と性能評価

遠隔ファイルアクセスに用いられる RPC をさらに細かく区分すると，同期型と非同期型の RPC がある．同期型 RPC とは，クライアントが 1 個ずつサーバにリクエストを送信し，完了を待ってから次に進む方式である．一方の非同期型は手順の進行状況に関係なくリクエストを発行することができる方式である．ここでは前者の同期型 RPC を対象としている．非同期型の RPC は回線遅延の影響を受けにくいという利点があるが，アプリケーションが非同期型の I/O に対応していなければ，遠隔ファイルアクセスの段階で予測などが必要となり，その効力を最大限に発揮することはできない．したがって，すべてのアプリケーションが非同期型の I/O に対応していない限りは，同期型 RPC による遠隔ファイルアクセスが必要でありその性能を高めなければならない．本論文において提案する手法は，あらゆる同期型 RPC を用いた遠隔ファイルアクセスに適用可能である．

本章では，複数のアクセスパターンと回線遅延における性能評価の結果を示す．

3.1　同期型 RPC によるファイルアクセス

図 2 に同期型 RPC によるファイルアクセスの流れを示す．これは，クライアントとサーバの 2 台の間でどのような情報がやりとりされるのかを示しており，クライアントの要求に含まれる fd はファイルの識別子，offset は要求データのファイル中における位置，size は要求データ量である．これら 3 つの引数を含む要求を受け取ったファイルサーバは，対応するデータをクライアントに返送する．この一連の動作を 1 回ずつ行うのが同期型 RPC によ

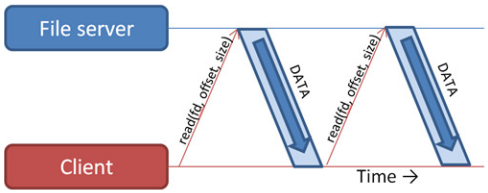


図 2　同期型 RPC による遠隔ファイルアクセス
Fig. 2　Remote file access using synchronous RPC.

る遠隔ファイルアクセスである．この場合，回線遅延が発生すると，クライアントの要求がファイルサーバに届くまでの時間が増大し，結果的に RPC の実行時間が増大する．この結果，単位時間あたりに実行される RPC の回数が減少し，性能が低下する．

3.2　評価するアクセスパターン

本論文ではシーケンシャルアクセス・ストライドアクセス・ランダムアクセス 3 つのアクセスパターンについて評価を行った．シーケンシャルアクセスは，ファイルの先頭から後方に向かって連続的にアクセスするパターンで，ストライドアクセスは一定量のアクセスとシークを繰り返し，ランダムアクセスはアクセスをファイル中のランダムな位置に行うものである．アプリケーションがファイルに対して行うアクセスは，一般的にこの 3 つに分類することができるため，これらのパターンおよび複数のアクセスパターンを混在させた条件で性能評価を行うことは，最適化などの効果を調べるうえで有用である．本論文においても，これらのアクセスパターンを基本として性能評価を行った．本研究が対象とする同期型 RPC によるファイルアクセスでは，特にストライドアクセスにおいて，アクセス量とシーク幅およびネットワーク環境によって最適値が存在し，これを求めることは非常に有意義である．また，アクセスパターンの変化に対応することも重要であり，本研究ではその点についても評価を行った．

3.3　実験環境および評価方法

本論文に示す評価に用いた実験環境は，Linux をセットアップした 2 台のコンピュータを GigabitEthernet で接続したものである．それぞれをクライアントとファイルサーバとして使用した．サーバ側には SATA 接続 7,200 回転のハードディスクを RAID-0 構成で 2 台接続しており，170 MB/s とネットワークに比べて十分高い性能が出ることを確認している．また，評価にあたっては，ソケットを直接用いたクライアント用およびサーバ用のプログラムを作成し，実際に同期型 RPC によるファイルアクセスを行った．回線遅延は任意のタイミングで測定することが可能であり，これにより遅延が変化する場合にも対応できるようになっている．

このプログラムは 3.1 節で述べた手順を忠実に実行する．回線遅延は Linux の tc コマンドによりクライアントコンピュータのネットワークインタフェースで発生させた．各評価結果については，3 回測定して平均値をとった値を示している．

以後，本論文における評価はすべて同じ方法で行っている．

3.4　RPC バッファサイズと性能

3.1 節で述べた同期型 RPC によってクライアントはファイルサーバにデータを要求する．この RPC には size というパラメータがあり，これは 1 回の RPC で転送するデータ量を表している．以後この値を RPC バッファサイズと呼ぶ．NFS や Gfarm などの既存のシステムでは RPC バッファサイズは固定値である．遠隔ファイルアクセスは，RPC バッファサイズ分のデータのやりとりを繰り返すことで行われる．

したがって，同期型 RPC でバッファサイズが固定値の場合，クライアントとサーバの間の回線遅延が増大することによって応答待ち時間の割合が増大すると，単位時間あたりに送受信できるデータ量が低下する．これはシーケンシャルアクセスを行う場合に問題となる．一方で，クライアントの要求がたとえ 1 バイトであったとしても，1 度の RPC ではバッファサイズ分の転送を行うため，RPC バッファサイズが必要とする読み込み量に対して大きい場合には，無駄な転送が増えて性能が低下する．この状態はランダムアクセスやストライドアクセスによって引き起こされる．

RPC バッファサイズが遠隔ファイルアクセスに与える影響を評価するため，様々な回線遅延やアクセスパターンのもとで RPC バッファサイズを変えながら性能評価を行った．以降にその結果を示す．

図 3〜図 7 はそれぞれ，シーケンシャルアクセス，512 KB の読み込みと 3 MB のシークを繰り返すストライドアクセス，同じく 3 MB の読み込みと 6 MB のシークのストライドアクセス，同 256 MB の読み込みと 768 MB のシークのストライドアクセス，そして 4 GB のファイル中のランダムな位置において 8 MB 読み込むランダムアクセスの性能を示している．横軸は RPC バッファサイズで 64 KB から 512 MB までの幅であり，縦軸はスループット［MB/s］である．グラフの各線は設定したネットワークの遅延時間で，0 ms から 100 ms の幅である．グラフ中のエラーバーは 3 回測定する際に得られた最大値と最小値を表している．

図 3 から分かるように，シーケンシャルアクセスにおいては回線遅延が小さい場合には RPC バッファサイズが約 1 MB 以上であれば十分な性能を示す．これ以下のサイズにおける性能低下は，プロトコルオーバヘッドなどの増大が原因である．一方，回線遅延が大きい場合，RPC バッファサイズも大きくなければ高い性能が出ない．たとえば 100 ms の回線遅延の場合，RPC バッファサイズが 4 MB でようやく 20 MB/s を超え，512 MB で 100 MB/s に達して飽和する．この特性には 2 つの原因があげられる．1 つは同期型 RPC が完了するまでの時間が，回線遅延が長くなるとその分だけ増大することである．時間がかかるほど次のリクエストを発行するまでに間隔が空き，何もしない時間が増える．2 つめは TCP の特性で，小さなサイズの転送を断続的に繰り返しても，輻輳ウィンドウのサイズが広がらず転

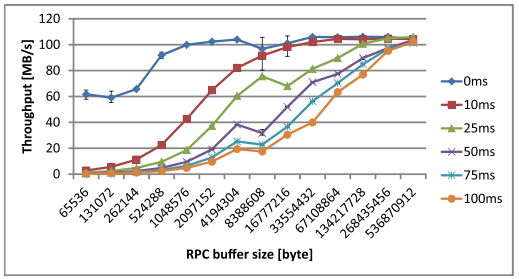


図 3　シーケンシャルアクセスの性能
Fig. 3　Performance of sequential access.

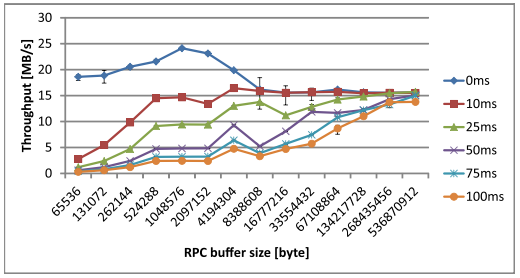


図 4　ストライドアクセス：512 KB 読み込み 3.5 MB シークの性能
Fig. 4　Performance of stride access: read 512 KB and seek 3.5 MB.

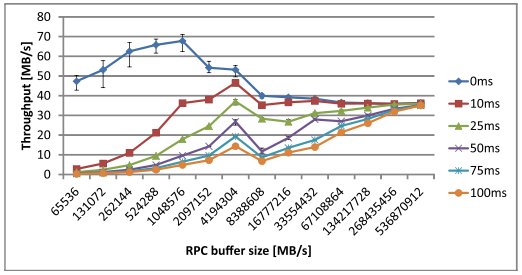


図 5　ストライドアクセス：3 MB 読み込み 6 MB シークの性能
Fig. 5　Performance of stride access: read 3 MB and seek 6 MB.

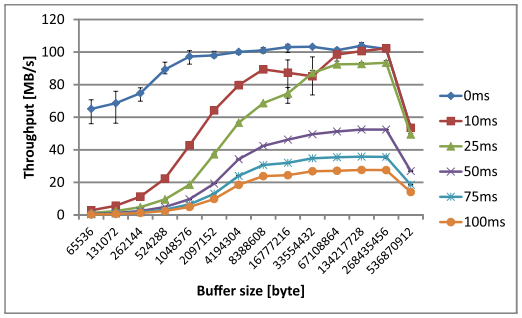


図 6　ストライドアクセス：256 MB 読み込み 768 MB シークの性能
Fig. 6　Performance of stride access: read 256 MB and seek 768 MB.

送速度が上がらないことである．したがって，シーケンシャルアクセスでは RPC バッファサイズを大きくするほど高い性能を出せる．512 MB 以上については，転送が数秒以上の長い時間を要することから，GigabitEthernet では性能の向上が見込めない．

一方，**図 4** と**図 5** のストライドアクセスから分かることは，回線遅延が小さい場合には RPC バッファサイズも小さい方が性能が高く，回線遅延が大きい場合には逆に RPC バッファサイズが大きい方が高い性能を示していることである．すなわち，アクセスパターンが同じであっても，回線遅延によって最適値が異なる．この性質は 3.1 節で述べたことが原因である．まず，RPC バッファサイズがクライアントの要求するデータ量に比べて大きすぎる場合には，ネットワーク上を転送したデータのうち使用されない部分が生じる．したがって，RPC バッファサイズを小さくすることで必要なデータのみを転送することが可能となり，高い性能を得ることができる．これは図 5 中 0 ms や 10 ms の線からも分かるように，ピーク性能が RPC バッファサイズの小さな領域に存在する．反対に，回線遅延が大きくなると小さな RPC バッファサイズでは十分な性能が出ない．これは，RPC の完了時間は最低でも回線遅延時間分かかることが原因である．したがって，回線遅延が大きい場合には，無駄な伝送を減らすために少量のデータを何回も転送するよりも，RPC バッファサイズを大きくして RPC 発行回数を抑えた方が高い性能を得られる．

図 6 のストライドアクセスはシーク幅が 768 MB と，回線帯域と比べて大きい．このような場合には，無駄な転送を減らすことのメリットが大きくなり，RPC バッファサイズは読み込みサイズに近い方が高い性能が出る．ランダムアクセスである**図 7** も同様で，読み込みサイズと等しい RPC バッファサイズの場合に高い性能を示している．

つまり，シーケンシャルアクセスならば RPC バッファサイズが大きい方が性能が高いが，

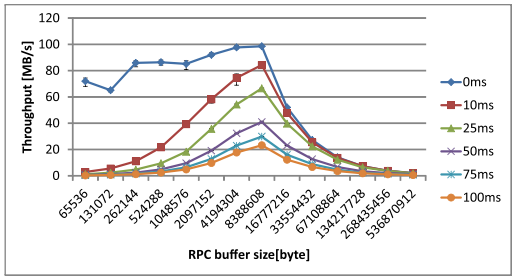


図 7　ランダムアクセス：4 GB のファイル中で 8 MB をランダムに読み込み
Fig. 7　Performance of random access: read 8 MB in a 4 GB file.

ストライドアクセスやランダムアクセスの場合は，シーク幅や読み込まれる量によって最適値が異なることが分かる．アクセスパターンの種類だけで一概に決めることができない点が問題である．

3.5　RPC バッファの作用

RPC バッファは，取得したデータを一時的に溜めておくものであるため，ある種のキャッシュと見なすこともできる．関連研究であげた AFS や Coda といったファイルシステムは，ネットワークの遅延や切断に備えるため，ファイル単位でいったんキャッシュしてからアクセスする．非常に大きな RPC バッファは，ファイル全体をキャッシュするのと似た状況になることがある．

しかしながら，最初からファイル全体を転送してキャッシュする方式は，単にバルク転送を行うだけであるから本提案手法を適用することはできない．また，本提案手法はクライアントとサーバが随時通信してファイルにアクセスすることを想定しており，ネットワークの切断に関しては想定していない．ネットワークの遅延に対処するためには，RPC バッファサイズを拡大してキャッシュに似た効果を出すことも考えられる．一方で副作用として，ファイル中のごくわずかな領域のみが必要な場合などでは，転送時間が非常に長くかかってしまう可能性がある．本提案手法は，このように対処が状況に応じて変わる場合に対して，動的にサイズ可変のキャッシュを用いて対応するものであるととらえられる．

4.　アクセスパターンの認識と RPC バッファサイズの動的変更

4.1　アクセスパターンの認識——RPC バッファ利用率

これまでにアクセスパターン・回線遅延・RPC バッファサイズの 3 要素が，遠隔ファイルアクセスの性能に与える影響を示してきた．最適な RPC バッファサイズを求めるためには，アクセスパターンの認識をする必要があり，本章ではそのための手法について述べる．

3 章に示したとおり，遠隔ファイルアクセスでは，クライアントが発行した RPC によってバッファサイズ分のデータを転送する．そこで，アクセスパターンを判断するにあたっては，この転送されたデータのうち実際にクライアントが必要としていた割合を測定することでアクセスパターンを判断することとした．この結果を RPC バッファ利用率と呼ぶこととする．

たとえば，シーケンシャルアクセスを行った場合には，RPC バッファの領域を連続して読み込むことを繰り返すため，RPC バッファ利用率は 100%になる．一方で 1 度の読み込みサイズ（ブロックサイズ）が RPC バッファサイズよりも小さなランダムアクセスを行った場合には，転送されたバッファのうち一部分のみしか利用されないことから，RPC バッファ利用率は 100%未満の値となる．したがって，RPC 利用率が高ければシーケンシャルアクセスであり，低ければランダムアクセスであると判断できる．

この RPC バッファ利用率を測定する方法はいくつか考えられる．最も直接的な方法は，クライアントアプリケーションのアクセスログを定期的に集計し，RPC バッファのうちどの程度使用されたかを集計する方法である．しかし，この方法では非常に小さなサイズの読み込みが何度も行われた場合に，ログサイズが増大して処理に要する時間が長くなるという問題がある．また，アクセス順序や回数は不要な情報であり，メモリを無駄にしてしまう．

そこで，本研究では，RPC バッファ中でどこが使われたかを示す情報を保持する手法を提案する．まず考えられるのは，RPC バッファ中のバイトごとに利用の有無を記録する方法で，利用されたバイト数を総バイト数で除算することで RPC バッファ利用率を求めることができる．ところがバイトごとに利用状況を管理した場合，1 バイトにつき 1 ビットの管理情報が必要となり，RPC バッファサイズの増加につれてメモリ使用量も増大してしまう．この問題を解決するために，本研究では RPC バッファ領域を n 個のブロックに分割して管理することとした．あるブロックに含まれる領域を 1 度でもクライアントアプリケーションがアクセスした場合に，利用済みとしてマークし，マークされたブロック数によって RPC バッファ利用率を定義する．

図 8 は RPC バッファをブロックに分割して RPC バッファ利用率を測定する手順を示している．図中 1 段目はアプリケーションが発行した I/O リクエストである．また，2 段目は遠隔アクセスのクライアントプログラムがサーバから受け取ったデータを保存している RPC バッファ領域を示しており，水色の部分はクライアントアプリケーションが読み込み

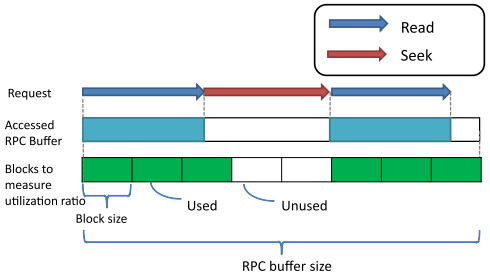


図 8　RPC バッファ利用率の測定手順
Fig. 8　Procedure to measure RPC buffer utilization ratio.

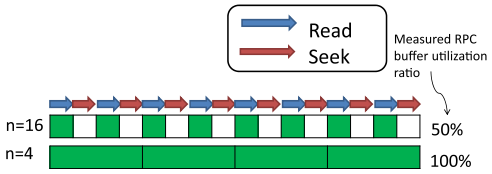


図 9　n の違いによる RPC バッファ利用率の変化
Fig. 9　Deference of RPC utilization ratio by n.

を行った領域である．3 段目は，この領域をブロックに分割し，その領域に含まれる RPC バッファが一部でも読み込まれていれば使用済みとして緑色にマークしたものである．3 段目に示すように，利用の有無をブロック数単位で RPC バッファ利用率を計算し，その式は以下のとおりとなる．

$$\text{RPC バッファ利用率} = \frac{\text{使用済みブロック数}}{n} \tag{1}$$

次に，ブロック分割数の n について考える．n を増加させると，バッファ領域の利用の有無を細かく管理できるようになるので，測定された RPC バッファ利用率がより正確になる．しかしながら，n を増加させて正確にすればするほど良いとは限らない場合がある．

そこで n が変わると RPC バッファ利用率の測定結果がどのように変化するかを**図 9** に示す．最上段はクライアントからのリクエストを表しており，同じ量の読み込みとシークを繰り返している．n が 16 の場合にはクライアントからのリクエストどおり，RPC バッファ利用率は 50%と測定され，ランダムアクセスであると判断される．一方で n が 4 の場合には RPC バッファ利用率は 100%と測定される．すなわち，シーケンシャルアクセスであると判断されるのである．RPC バッファ利用率の定義として，あるブロックが 1 度でもアクセスされれば使用済みとしていることから，n を減らすと，測定される利用率は変化しないか増加するかのどちらかである．極端な例として n が 1 ならば，どのようなアクセスも 100%と測定される．

この性質をうまく利用することで，RPC バッファサイズをどう変化させるべきかが回線遅延の大小によって変わる場面において，適切に対応する手法を考案した．これは後ろに述べる帯域幅遅延積が重要な数字となる．

前章で述べたように，低遅延環境下のランダムアクセスやストライドアクセスにおいて，RPC バッファサイズを小さくする意義は，無駄な転送を減らすことである．しかし，RPC バッファサイズを小さくすると必然的に RPC 発行回数が増える．RPC の完了時間は最低でも回線遅延分を必要であるから，回線遅延が増大すればそれだけ時間がかかる．したがって，RPC バッファサイズを小さくするメリットは，削減できた無駄な転送量が，回線遅延分の時間に転送できるデータ量を上回っている間にのみ存在する．つまり，このメリットが失われた状況では，シーケンシャルアクセスと同じ扱いをして，まとめて転送してしまった方がよい．このアクセス方法は，data sieving [12] と呼ばれている．

この点を n の決め方に応用することを考える．回線遅延分の間に転送できるデータ量は，回線遅延時間と帯域の積で表すことができ，以後これを帯域幅遅延積と呼ぶ．そして，この帯域幅遅延積がシーク量，要するに削減しうる無駄な転送よりも大きければ，そのシークされた領域を利用済みと見なすことでシーケンシャルアクセスと同じ扱いをできる．これを実現するためには，RPC バッファ利用率測定に用いるブロックサイズが帯域幅遅延積になるよう，n を設定すればよい．したがって，n は次の式を満たす値をとればよい．

$$\text{帯域幅遅延積} = \text{ブロックサイズ} \tag{2}$$

$$= \frac{\text{RPC バッファサイズ}}{n} \tag{3}$$

この n と，測定された回線帯域・回線遅延をもとに RPC バッファ利用率を求める．

4.2　RPC バッファサイズの動的変更

4.1 節に示した方法で RPC バッファ利用率を求め，アクセスパターンを判別することができる．RPC バッファ利用率は，連続的なアクセスでは高く，不連続なアクセスでは低くなる．したがって，RPC バッファ利用率が高ければシーケンシャルアクセスであり，低ければランダムアクセスであると判断することができる．この判断に基づき，1 度の RPC でやりとりするデータ量である RPC バッファサイズを動的に変更したい．この動的な変更を

行うにあたっては，一時的なアクセスパターンの変化にすぐ対応するか，あるいはある程度継続したアクセスパターンを重視するかを設定できるようにした．そのために，過去 m 回分の RPC バッファ利用率の平均値に基づいて RPC バッファ利用率を変更することとした．また，バッファサイズが非常に大きい場合，1 度の RPC で必要とする転送時間が長く，最適ではない状態が長く続くとペナルティが大きい．そのため m の値について，RPC バッファサイズが *LARGE_BUFSIZE* を上回ったら M_{small}，*LARGE_BUFSIZE* 以下であれば M_{normal} と設定する．アクセスパターンの変化の頻度やアプリケーションのアクセスの性質によりこれらのパラメータの最適値は変わる．

この RPC バッファ利用率は RPC バッファのうちどれだけのデータを転送すべきであったかを示す数値であるから，利用率が低ければ RPC バッファサイズを縮小し，逆に高ければ RPC バッファサイズをある増加率で乗算して拡張する．利用率の高低の判断は 2 つの閾値によって行い，高い方が U_{high}，低い方を U_{low} とする．RPC バッファサイズの変更手順を，以下に示す．

if (RPC バッファ利用率 $\rangle$ U_{high})
　RPC バッファサイズ　*=
　　RPC バッファサイズ増加率
else if (RPC バッファ利用率 $\langle = U_{low}$)
　RPC バッファサイズ *=
　　RPC バッファ利用率

この手順により，シーケンシャルアクセスによって RPC バッファ利用率が高い状態が続けば RPC バッファサイズが増加する．RPC バッファサイズ増加率については，RPC バッファ利用率が高いことからは最適値との乖離量を知ることはできないため，RPC バッファサイズのとりうる値のうち，最小限の増加となる値をとる．一方でランダムアクセスによって RPC バッファ利用率が低くなると，RPC バッファサイズが縮小される．新しい RPC バッファサイズは現在の RPC バッファサイズと利用率を乗算した値であることから，最適値との乖離が大きいほど素早く縮小される．その後，RPC バッファサイズが縮小されて最適値に達すると利用率が高い値になるが，これはシーケンシャルアクセスではないので RPC バッファサイズを拡大すべきではない．しかしこの手順だけでは，利用率の上昇によりバッファサイズが増加し，最適値を外れて利用率が低下してバッファサイズが元の小さなサイズに戻るというサイクルを繰り返してしまう．ただし，シーケンシャルアクセスに移行していた場合にはそのままバッファサイズが増大し，それは正しい動作である．

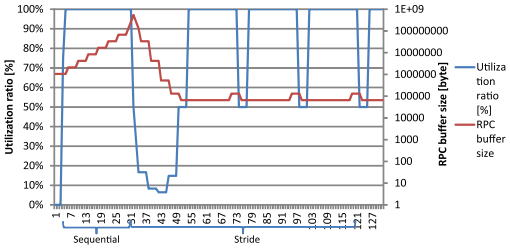


図 10　提案手法の挙動
Fig. 10　Behavior of proposed method.

しかしながら，RPC バッファサイズが最適値を頻繁に外れることは好ましくなく性能に悪影響を与える．そこで，RPC バッファサイズ増加直後に利用率が低下した場合，以降 p 回の RPC 発行の間はバッファサイズを拡大しない仕組みを導入した．p が大きければランダムアクセスの継続性を重視し，小さければアクセスパターンの変化に素早く反応できるようになる．

また，RPC バッファサイズについては *MAX_BUFSIZE* という上限を設けている．その理由は，メモリ量が有限であることや，シーケンシャルアクセスであっても，ある程度以上の RPC バッファサイズにおいては性能が飽和してしまうことである．本論文における性能評価で，性能が飽和することについては，図 3 からも分かり，512 MB 以上に増大させても性能が向上するとは見込めない．したがって，以降すべての評価について，*MAX_BUFSIZE* は 512 MB としている．

4.3　実装の検証

この章でこれまで述べてきた，RPC バッファ利用率に基づく RPC バッファサイズの動的変更手法を実装して動作させ，その挙動を検証した．

この検証にあたっては，各種パラメータを次のとおり設定した．*LARGE_BUFSIZE* を 128 MB，M_{small} を 1，M_{normal} を 4，U_{high} を 95%，U_{low} を 50%，RPC バッファ増加率を 2，p を 16 とした．また，RPC バッファサイズは 2 のべき乗をとるので，RPC バッファサイズを縮小する際には，RPC バッファ利用率が 50%以下であれば半分，25%以下では 4 分の 1，12.5%以下では 8 分の 1 とした．

図 10 に示したグラフは，1 GB のシーケンシャルアクセスから 61 KB の読み込みと 1 MB のシークを繰り返すストライドアクセスに移行した際の RPC バッファ利用率と RPC バッ

ファサイズの推移を示している．横軸は RPC 発行回数で，縦軸は左が RPC バッファ利用率［%］，右が RPC バッファサイズ［KB］の対数をとったものである．

RPC30 回目まではシーケンシャルアクセスが行われたことで RPC バッファ利用率が上昇し，それにともなってバッファサイズも拡大している．それ以降はストライドアクセスが行われたことで RPC バッファ利用率が低下し，バッファサイズが縮小する．バッファサイズが縮小し，適正値に近づくと徐々に利用率が向上し，適正値となる．バッファサイズが適正値になると利用率も 100%を示す．しかし，これはシーケンシャルアクセスではないため，バッファサイズを拡大すると利用率が低下する．そのため，前節で述べたように p 回分はバッファサイズを維持することで最適値をより長く保っている．この定常状態がグラフの後半に示されている．

4.4　パラメータの検証

前節ではいくつかのパラメータを固定していたが，これらの値が性能に与える影響について評価を行った．その内容と考察を示す．

まずは，パラメータ p について記す．この値は，RPC バッファサイズを拡大したにもかかわらず RPC バッファ利用率が低下してしまった場合，RPC バッファサイズを元に戻し，以後 p 回の RPC 中は RPC バッファサイズを拡大しないようにするものである．この値が非常に大きい場合，一定のアクセスパターンが継続する場合には性能が高く，変化する場合には低くなるなどの問題が生じる可能性がある．そこで，この p を変化させた場合の性能について評価を行った．評価は p を 2 から 32 の間で変化させ，それぞれにおいて性能を測定した．この評価もクライアントとサーバの 2 台の間におけるファイルアクセスであり，遅延は LAN 環境でありこれまでの評価中の 0 ms と同等である．

まずは 512 KB の読み込みと 3 MB のシークを繰り返すストライドアクセスの性能を**図 11** に示す．性能の変動幅は最小が $p = 2$ の 33 MB/s で，最大が $p = 16$ の 35.5 MB/s と，10%以内であった．元々，予備評価図 4 などからも分かるように，最適値に隣接するバッファサイズを選択しても，性能の低下幅はそれほど大きくはない．そのうえ，たとえば p が 4 ならば，最適値以外の RPC バッファサイズが利用されるのは RPC5 回につき 1 回であり，さらに影響は低減される．したがって，変化しないアクセスパターンにおいては p の影響は小さく，この中で評価した範囲においては，過大または過小といった問題は見られない．なお，シーケンシャルアクセスなど，RPC バッファサイズが最大値まで拡大される状況では，p はまったく影響を及ぼさない．

次に，アクセスパターンが変化する場合について，p の影響を評価した．**図 12** は 25 MB のシーケンシャルアクセスと，512 KB 読み込み 1.5 MB シークのストライドアクセス 50 回を交互に繰り返す混在アクセスパターンにおける性能である．性能は 30 MB/s から 31.8 MB/s の間に収まっており，ここでも p は性能に大きな影響を与えていないことが分かる．

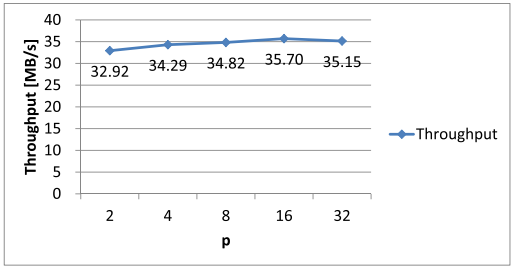


図 11　512 KB 読み込み 3 MB シークのストライドアクセスにおける p の影響
Fig. 11　Effect of the p on the stride access (read 512 KB and seek 3 MB).

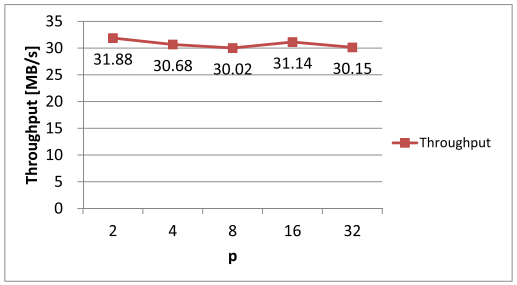


図 12　混在アクセスパターンおける p の影響
Fig. 12　Effect of the p on the mixed access patterns.

したがって，アクセスパターンが一定であっても変化する場合であっても，パラメータ p は性能に対し，顕著な影響を与えていないことが分かる．

次に，*LARGE_BUFSIZE* について検討する．*LARGE_BUFSIZE* はその意味合いからも，転送に長時間かかるような RPC バッファサイズを示す．サーバ・クライアント間で転送時間を無視した場合，数十から数百ミリ秒オーダの遅延時間では 1 秒間に数回から数十回の RPC 実行が可能である．したがって，長い転送時間を要することによって RPC 実行回数/秒が 1 を下回る場合，その RPC バッファサイズは相対的に大きなものであるといえる．そして，本提案手法は RPC 実行ごとに評価を行うことから，アダプティブな対応を行

ううえで，転送に秒オーダの時間がかかるサイズを *LARGE_BUFSIZE* とするのが妥当である．これよりも大きな RPC バッファサイズでは利用率の平滑化による安定よりも，素早く対応することの方が重要である．したがって，Gigabit Ethernet の場合，その 1 秒間の最大転送量である 128 MB を 1 つの目安とし，ここではその値を用いている．

RPC バッファの利用率の平均数である M_{normal} については，特に RPC バッファサイズが小さい場合に重要である．なぜならば，RPC バッファ利用率を測定する対象が小さく，測定結果が揺れる可能性が高いからである．または，帯域幅遅延積が大きく，利用率測定のブロック分割数がきわめて小さい場合にも同様である．立ち上がり時などではブロック分割数が 1 となる場合があり，このような場合には，2 以下の平均によって RPC バッファサイズを求めてしまうと適切に RPC バッファサイズの変更ができない．なぜならば，2 以下の場合 RPC バッファ利用率は 0%・50%・100%の 3 値のみをとることになり，RPC バッファサイズを 2 の倍数としている本実装においては，大きくするか小さくするかの二択に限られてしまうからである．RPC バッファサイズの拡大・縮小・維持の 3 つの閾値にフィットしうる最小の M は 4 であり，今回はその値を M_{normal} としている．M_{small} については *LARGE_BUFSIZE* からも分かるように，回線遅延などと比較して 1 回の RPC で生じる転送時間のオーダが大きい場合であり，素早い適応のために極力小さな値をとる必要がある．*LARGE_BUFSIZE* を超えるような RPC バッファサイズの場合，RPC バッファ利用率測定のためのブロック数が少なくなりにくいことと，アクセスパターンの変化に素早く対応する必要性から M_{small} を 1，すなわち平均値を用いないことも十分に考えられる．本論文における評価においても，この理由から M_{small} を 1 とした．

また，利用率の閾値である U_{high} については，シーケンシャルアクセスまたはそれに準ずる（data sieving）状況であることが分かればよく，それは RPC バッファ利用率の定義からも 100%に近い値である．ただし，U_{high} の影響はあまり大きくない．このことは，p の影響の小ささから導くことができる．p の検証で示したように，隣接する RPC バッファサイズでは極端に性能が変化しない場合が多く，1 度 RPC バッファサイズを大きくしてから戻すという操作を繰り返したとしても p を小さくすることと変わりないからである．U_{low} については，RPC バッファサイズがとりうる値に依存している．本論文の実装では 2 の倍数であるから，RPC バッファ利用率が 50%を下回らなければ，手順上 RPC バッファサイズを小さくすることができない．したがって，ここでは U_{low} は 50%としている．

5. 性能評価

本提案手法を評価環境に実装し，性能評価を行った．実際のアプリケーションを用いて評価する方法もあるが，実験に用いている実装が汎用のファイルシステムとして利用できるだけの機能やインタフェースを備えていない．そこで，複数のアクセスパターンおよびそれらが混在するパターンについての評価を行い，より実際の利用に即する内容とした．評価環境は 3.3 節に示したとおりである．

5.1 シーケンシャルアクセス

図 13 はシーケンシャルアクセスの性能比較のグラフで，横軸は回線遅延，縦軸はスループットを示している．グラフ中の各項目で，Optimal は 3.4 節に示した各グラフ中で，最も高い性能を示した部分を抽出したもので，性能の限界値を示している．Adaptive は本提案手法を適用した場合の性能である．1 MB fixed は Gfarm バージョン 2.4.1 相当の性能を示しており，既存システムの一例で比較対象でもある．性能測定はいずれも 6 GB のファイル転送によって行った．

グラフからも分かるように，1 MB fixed では遅延が増大するに従って性能が大幅に低下するが，Adaptive ではそれ程低下せず，高い性能を保っている．Adaptive と Optimal との乖離は回線遅延の増大につれて大きくなるが，これは RPC バッファサイズが立ち上がるまでに時間がかかっていることが原因で，より大きなファイルを転送すると乖離は小さくなる．**表 1** に示したように，提案手法が選択した RPC バッファサイズは，低遅延環境においてバッファサイズが飽和する場合を除いては，実測パフォーマンスから求めた最適値

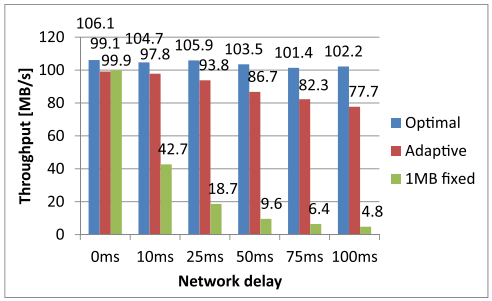


図 13　シーケンシャルアクセスの性能比較
Fig. 13　Comparison of sequential access performance.

表 1　シーケンシャルアクセスで選択された RPC バッファサイズ
Table 1　Selected RPC buffer size in sequential access.

Network delay	Adaptive	Optimal
0 ms	512 MB	128 MB
10 ms	512 MB	256 MB
25 ms	512 MB	512 MB
50 ms	512 MB	512 MB
75 ms	512 MB	512 MB
100 ms	512 MB	512 MB

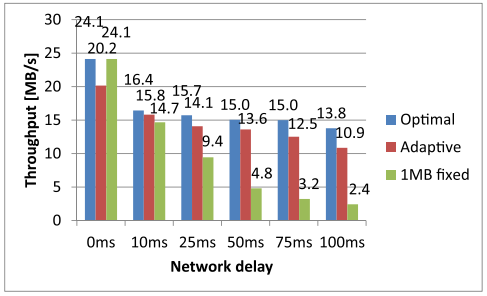


図 14　ストライドアクセス（512 KB 読み込み 3 MB シーク）の性能比較
Fig. 14　Performance comparison of stride access (read 512 KB and 3 MB seek).

（Optimal）と同等となっている．

5.2　ストライドアクセス

図 14 と**図 15**，**図 16** はそれぞれ 512 KB 読み込み 3 MB シークを繰り返すストライドアクセスと，同 3 MB 読み込み 6 MB シーク，同 256 MB 読み込み 768 MB シークの性能比較を行ったグラフである．グラフの軸や各項目はシーケンシャルアクセスと同じである．ほとんどのケースで，提案手法は 1 MB fixed よりも高い性能を示しており，Optimal に近づいていることが分かる．

表 2 と**表 3**，**表 4** に，各回線遅延の下で本提案手法が選択した RPC バッファサイズと Optimal の値を示す．こちらも，本提案手法が多くの状況で最適な RPC バッファサイズかそれに近い値を選択していることが読み取れる．

図 17 は −2.5 MB から +5.5 MB の幅でランダムにシークを行い，512 KB の読み込みを行うストライドアクセスの性能比較である．この場合，ファイルの先頭に戻ることがあ

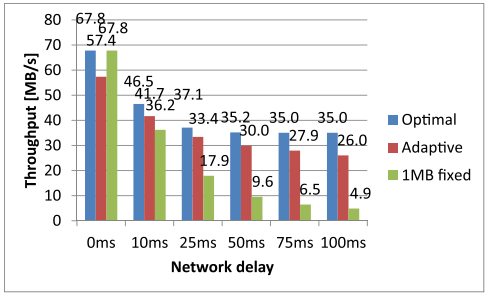


図 15　ストライドアクセス（3 MB 読み込み 6 MB シーク）の性能比較
Fig. 15　Performance comparison of stride access (read 3 MB and 6 MB seek).

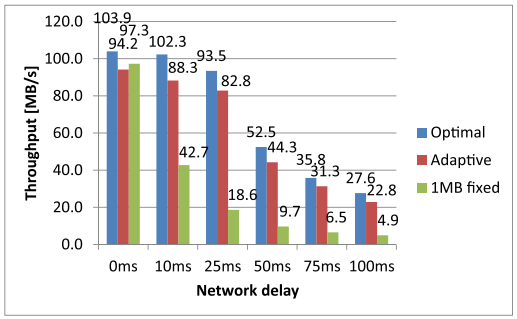


図 16　ストライドアクセス（256 MB 読み込み 768 MB シーク）の性能比較
Fig. 16　Performance comparison of stride access (read 256 MB and 768 MB seek).

表 2　ストライドアクセス（512 KB 読み込み 3 MB シーク）で選択された RPC バッファサイズ
Table 2　Selected RPC buffer size in stride access (read 512 KB and seek 3 MB).

Network delay	Adaptive	Optimal
0 ms	512 KB	1 MB
10 ms	4 MB	4 MB
25 ms	512 MB	512 MB
50 ms	512 MB	512 MB
75 ms	512 MB	512 MB
100 ms	512 MB	512 MB

表 3　ストライドアクセス（3 MB 読み込み 6 MB シーク）で選択された RPC バッファサイズ
Table 3　Selected RPC buffer size in stride access (read 3 MB and seek 6 MB).

Network delay	Adaptive	Optimal
0 ms	1 MB	1 MB
10 ms	4 MB	4 MB
25 ms	4 MB	4 MB
50 ms	512 MB	512 MB
75 ms	512 MB	512 MB
100 ms	512 MB	512 MB

表 4　ストライドアクセス（256 MB 読み込み 768 MB シーク）で選択された RPC バッファサイズ
Table 4　Selected RPC buffer size in stride access (read 256 MB and seek 768 MB).

Network delay	Adaptive	Optimal
0 ms	128 MB	128 MB
10 ms	128 MB	256 MB
25 ms	256 MB	256 MB
50 ms	256 MB	128 MB
75 ms	256 MB	128 MB
100 ms	256 MB	128 MB

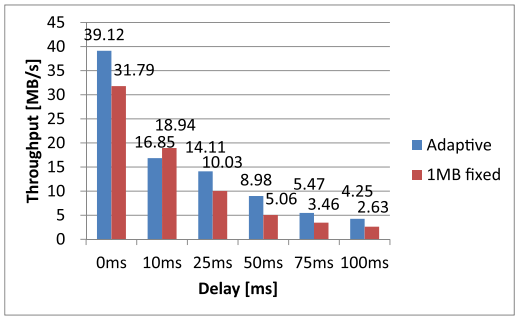


図 17　ランダムシーク幅（−2.5 MB から +5.5 MB）512 KB 読み込みストライドアクセスの性能比較
Fig. 17　Performance comparison of stride access (read 512 KB and −2.5 MB – +5.5 MB random seek).

り，3.5 節に述べたキャッシュの作用が働くことがある．シーク幅が変動することから一定の RPC バッファサイズをとらないが，1 MB fixed よりも高い性能を示していることが分かる．

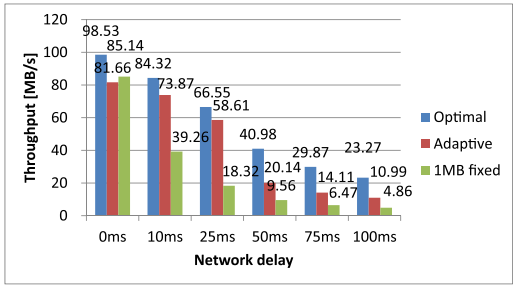


図 18　4 GB ファイル中の 8 MB ブロックランダムリードの性能比較
Fig. 18　Performance comparison of random access (block size = 4 MB, file size = 4 GB).

表 5　4 GB ファイル中の 8 MB ランダム読み込みで選択された RPC バッファサイズ
Table 5　Selected RPC buffer size in random read access (4 GB file, 8 MB block size).

Network delay	Adaptive	Optimal
0 ms	8 MB	8 MB
10 ms	8 MB	8 MB
25 ms	8 MB	8 MB
50 ms	16 MB	8 MB
75 ms	16 MB	8 MB
100 ms	16 MB	8 MB

5.3　ランダムアクセス

図 18 は 4 GB のファイル中でランダムな位置の 8 MB 読み込みを行った場合の性能である．この場合においても，1 MB fixed と比較して一定の性能向上が見られる．**表 5** から分かるように，いずれの場合も Optimal と同じか近い値を選択している．

5.4　混在パターン

図 19 は，50 MB のシーケンシャルアクセスと，2 MB 読み込み 4 MB シークを 25 回繰り返すストライドアクセスを交互に行った混在アクセスパターンである．このような単一ではないアクセスパターンにおいても，本提案手法は性能向上を達成することができた．

5.5　提案手法のオーバヘッド

本提案手法を実装するにあたっては，RPC バッファの利用状況を管理するためのメモリと，履歴用のバッファが必要である．それぞれ，ブロック数を 4,096 とすると 512 バイト，倍精度浮動小数点の変数 8 個で 64 バイトとなり，合計 576 バイトにすぎず，通信用のバッファなどに比べれば非常に小さい．また，処理に必要な時間も非常に短く，実測で RPC1

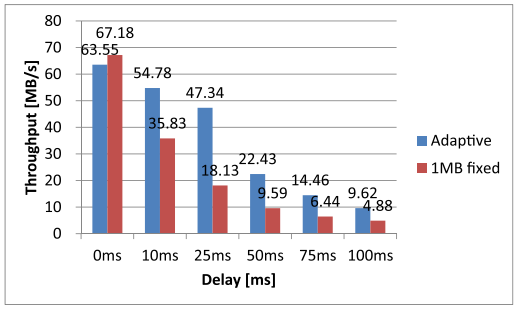


図 19　シーケンシャルアクセスとストライドアクセス混合の性能比較
Fig. 19　Performance comparison of mixture of sequential access and stride access.

回あたり 160 μ 秒であった．オーバヘッドが非常に小さいことから，本提案手法による性能向上をそのまま活かすことが可能である．

6. ま と め

本論文では，同期型 RPC を用いた遠隔ファイルアクセスについて，回線遅延と回線帯域，アクセスパターンの 3 つの要素を同時に考慮しながら最適化する方法を示した．最適化する対象は 1 度の RPC によって転送されるデータ量を表す RPC バッファサイズである．まずはこの値を変えながら様々な条件で予備評価を行い，最適値について調べた．その後に RPC バッファサイズを最適化するための手法について述べ，この手法が性能向上に有効であることを性能評価によって示した．提案手法は非常に小さなオーバヘッドで大きな性能向上を達成できており，多くの場合で最良のパラメータを選択していた．シンプルなアイディアで種々の条件を勘案して最適化できる点は本提案手法の最大の特長であり，あらゆるシステムに導入可能であることからも，非常に応用の範囲が広いといえる．

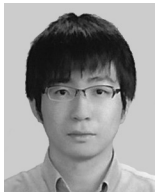
ここでは同期型の I/O と RPC を用いることを前提としていたが，今後は同期型 RPC だけではなく，非同期に先読みを行うなど，さらに性能を向上させることができる可能性のある方式についても検討を進める．さらに，分散環境において効率的な遠隔ファイルアクセスを行うことについては，今後研究すべき課題である．また，実システムへの導入を進め，より実践的な性能評価が必要である．

謝辞　本研究の一部は，JST CREST「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」および文科省次世代 IT 基盤構築のための研究開発「研究コミュニティ形成のための資源連携技術に関する研究」（データ共有技術に関する研究）による．

参 考 文 献

1) Chervenak, A.L., Foster, I.T., Kesselman, C., Salisbury, C. and Tuecke, S.: The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets, *Journal of Network and Computer Applications*, Vol.23, pp.187–200 (1999).
2) Tatebe, O., Hiraga, K. and Sod, N.: Gfarm Grid File System, *New Generation Computing*, Vol.28, No.3, pp.257–275, Ohmsha, Ltd. and Springer (2010).
3) Callaghan, B., Pawlowski, B. and Staubach, P.: NFS Version 3 Protocol Specification, RFC 1813 (1995).
4) Carns, P.H., Iii, Ross, R.B. and Thakur, R.: Pvfs: A parallel file system for linux clusters, *ALS 00: Proc. 4th annual Linux Showcase and Conference* (2000).
5) Braam, P.J.: Lustre, available from ⟨http://www.lustre.org/⟩.
6) Howard, J.H.: Scale and performance in a distributed file system, *ACM Trans. Computer Systems*, Vol.6, No.1, pp.51–81 (1988).
7) Satyanarayanan, M.: Coda: A Highly Available File System for a Distributed Workstation Environment, *IEEE Trans. Comput.*, Vol.39, No.4, pp.447–459 (1990).
8) Allcock, W.: GridFTP: Protocol Extensions to FTP for the Grid, *Global Grid Forum Draft* (2003).
9) 伊藤建志，大崎博之，今瀬　眞：データ転送プロトコル GridFTP のための並列 TCP コネクション数調整機構の性能評価，電子情報通信学会技術研究報告情報ネットワーク 105(628), pp.201–208 (2006).
10) Costa, L.B. and Ripeanu, M.: *Towards Automating the Configuration of a Distributed Storage System*, pp.201–208, IEEE (2010).
11) Madhyastha, T.M., Elford, C.L. and Reed, D.A.: Optimizing Input/Output Using Adaptive File System Policies, *Proc. 5th Goddard Conference on Mass Storage Systems and Technologies*, pp.493–514 (1996).
12) Thakur, R., Gropp, W. and Lusk, E.: Data Sieving and Collective I/O in ROMIO, *Proc. 7th Symposium on the Frontiers of Massively Parallel Computation*, pp.182–189 (1988).

(平成 23 年 1 月 28 日受付)
(平成 23 年 6 月　4 日採録)



大辻　弘貴（学生会員）

昭和 63 年生．平成 23 年筑波大学情報学群情報科学類卒業．現在，同大学大学院システム情報工学研究科在学中．分散ファイルシステムに興味を持つ．

建部　修見（正会員）

昭和 44 年生．平成 4 年東京大学理学部情報科学科卒業．平成 9 年同大学大学院理学系研究科情報科学専攻博士課程修了．同年電子技術総合研究所入所．平成 17 年独立行政法人産業技術総合研究所主任研究員．平成 18 年筑波大学大学院システム情報工学研究科准教授．博士（理学）（東京大学）．超高速計算システム，グリッドコンピューティング，並列分散システムソフトウェアの研究に従事．日本応用数理学会，ACM 各会員．