Vol.2011-HPC-130 No.30
2011/7/28

Non-blocking RPC を用い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]

これまで，同期型 RPC を用いた遠隔ファイルアクセスの最適化に取り組んできたが，本論文ではこれを非同期すなわちノンブロッキング RPC によって更に性能向上を図った．これによりバックグラウンドで転送を行う事が可能になる．その性能の変化について評価およびシミュレーションを行い，その結果を示す．また，シーケンシャルアクセスだけではなくストライドアクセスなどの非連続アクセスについても，読み込み位置の予測を行い，先回りして転送する手法を提案し，その有効性について検証する．これらを通じて，より高速な遠隔ファイルアクセスの実現に向けた手法を論じる．

1. 序 論

計算機で処理すべきデータは年々増加しており，効率的に扱うためには広域環境でデータを共有する必要がある．大容量のデータを取り扱うデータインテンシブコンピューティングや e-サイエンスの発展により，複数のスパコンセンターや地理的に離れた拠点においてデータの共同解析が行われる機会が増えている．このような事から，広域分散ファイルシステムが広く用いられている．データ共有およびアクセスの高速化のためにファイル複製[1] の手法が用いられる事があるが，データを利用するクライアントのストレージ容量が十分でない場合には複製を作る事が出来ない．または，ファイルのごく一部分のみが必要な場合であっても大量のデータ転送が発生し，非効率である．このような場合には，遠隔ファイルアクセスを行ってデータを転送する．しかしながら，ファイル複製時にはバルク転送が行われるのに対し，遠隔ファイルアクセスではクライアントが指定した場所を逐一転送するため，回線遅延の影響を大きく受ける．性能が低下した場合にはシステム管理者が設定を最適化することになるが，利用状況や環境を考慮しなければならないうえ，変化にもその都度対応しなければならないため，困難を伴う．そこで本論文では，分散ファイルシステム中において単一クライアント・サーバ間で行われる遠隔ファイルアクセスに注目し，その自動的な最適化について述べる．

†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
JST CREST

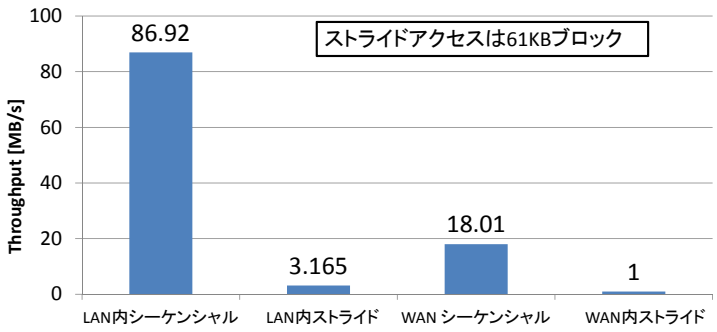

図 1 既存システム[2] における性能
LAN のネットワーク遅延は 50 μ s，WAN は 25ms
Fig. 1 Performance of existing method.
Network delay of LAN is 50us and WAN is 25ms.

2. 関 連 研 究

遠隔ファイルアクセスの方式は 2 つに大分される．一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の指定によって必要な部分を転送する方式と，Web(http) 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，予め設定した一定量のデータを一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 1MB 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] が挙げられる．本研究は前者の RPC による転送を行う形式を対象としている．

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 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

© 2011 Information Processing Society of Japan

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出来る．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．その他にも，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事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．9) は実際に計測されたスループットを元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．

本論文では，利用状況と環境に基づいて，人手を介さずに RPC ベースの遠隔ファイルアクセスを最適化する研究について示す．

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 を更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．本研究では後者の非同期型 RPC を取り入れて遠隔ファイルアクセスの最適化を行った．

アプリケーションは同期的に I/O リクエストを発行するものと仮定し，非同期 RPC による先読みなどを行い，性能を向上させた．本論文において提案する手法は，RPC を用いたあらゆる遠隔ファイルアクセスに適用可能な手法である．本章では，同期型・非同期型 RPC による性能の比較やアクセスパターンによる性能の変化について述べる．

3.1 RPC による遠隔ファイルアクセス

図 2 に本研究が対象とする遠隔ファイルアクセスの手順を示す．これらは，クライアントとサーバの間でどのようなリクエストとレスポンスがあるかを示した図である．クライアントはアプリケーションからの要求に基づいて，ファイル中で必要な位置とサイズをサーバに送信し，サーバはそのデータを返す．アプリケーションはその中からデータを読み込む．この一連の動作の繰り返しにより遠隔ファイルアクセスが行われる．サーバが返すデータ量をここでは RPC バッファサイズと呼ぶ．

同期型では RPC が完了してから次のリクエストを送信するのに対して，非同期型では完了を待たずに連続的にリクエストを送信することが出来る．**図 3** は同期型・非同期型それぞれにおける遠隔ファイルアクセスの動作を示している．

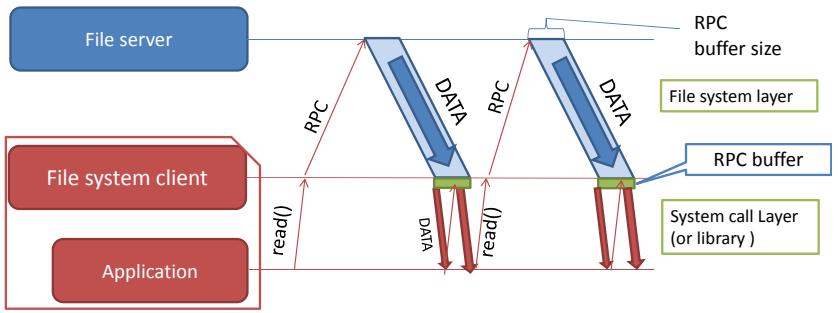

図 2 RPC による遠隔ファイルアクセス
Fig. 2 Remote file access with RPC

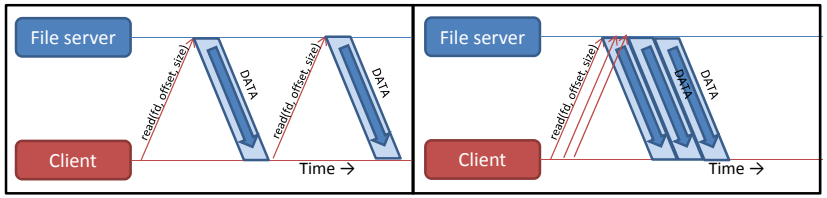

図 3 同期型・非同期型 RPC の比較（右側が非同期）
Fig. 3 Synchronous RPC(left) and Asynchronous RPC

RPC バッファサイズは一般的には固定であるが，筆者らがこれまで進めてきた研究では同期型 RPC において RPC バッファサイズの動的な最適化を行った．本論文では，非同期型 RPC を取り入れ，遠隔ファイルアクセスの最適化を行う．尚，ファイルシステムを利用するアプリケーションは同期 I/O を用いるものと仮定する．

3.2 回線遅延とアクセスパターンの影響

一般に，回線遅延が増大すると性能が低下する．それは，クライアントからのリクエストがサーバに届くまでの時間および，サーバからのレスポンスがクライアントに届くまでの時間が長くなるからである．同期型の RPC を用いている場合，この待ち時間の間は何も出来ない無駄な時間となる．一方で非同期型の RPC を用いた場合，同期型では何も出来ない間であっても，データの転送を行い，遅延の大きなネットワークでも効率的な遠隔ファイルアクセスを行う事が出来るようになる．

一方でアクセスパターンも性能を大きく左右する．例えば，ファイルを連続的にアクセス

© 2011 Information Processing Society of Japan

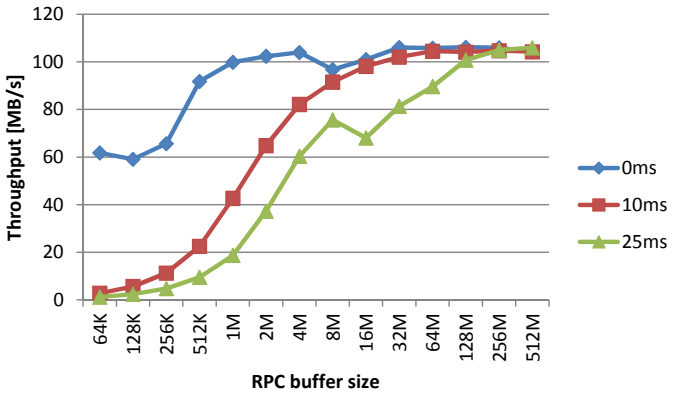

図 4 ブロッキング RPC によるシーケンシャルアクセスの性能
Fig. 4 Performance of sequential access with blocking RPC

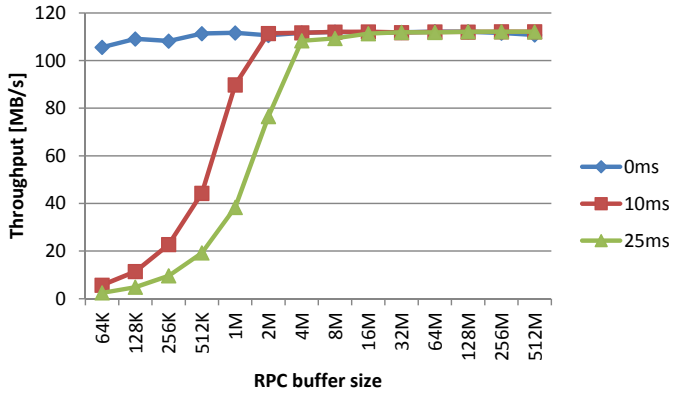

図 5 ノンブロッキング RPC によるシーケンシャルアクセスの性能
Fig. 5 Performance of sequential access with non-blocking RPC

するシーケンシャルアクセスでは，次々に転送するか予め非同期に先読みを行う等の方法により，転送速度を向上することができる．しかしながら，ランダムアクセスやストライドアクセスなど不連続な領域をアクセスする場合には，必要な領域をその都度転送したり，ある程度の無駄が生じることを前提に先読みを行う事になり，性能の低下が避けられない．

図 4 は同期型 RPC を用いた場合に，回線遅延と RPC バッファサイズの変化が，シーケンシャルアクセスの性能にどのような影響を与えるかを示したものである．この図から分かるように，遅延が増大すると性能が低下する．この場合には一度のリクエストで転送する量，すなわち RPC バッファサイズを大きくすることで性能が向上することが分かる．一方で，**図 5** は非同期型 RPC を用いた場合の性能である．この場合においても，RPC バッファサイズが小さい場合には回線遅延の影響を受けているが，同期型 RPC と比べるとあまり大きくは無い．これは，非同期型の場合には RPC の応答待ち時間を利用して，バックグラウンドで転送を行えるからである．また，連続的に転送が行えることにより，TCP のウィンドウサイズが縮小されないことも性能の向上に寄与している．

次に，非連続のアクセスパターンについて検討する．シーケンシャルアクセスではファイルに対して連続的なアクセスを行ったため，RPC バッファサイズを大きくしたり非同期転送を行ったりすることによってスループットを向上することが出来たが，次にアクセスされる場所が分からない状況では，必ずしも前もって転送することが良いとは限らない．まずは，ストライドアクセスの場合について述べる．ストライドアクセスとは，一定の間隔のシークとアクセスを繰り返すアクセスパターンである．**図 6** に 2 種類のストライドアクセスの例を示す．1 つめは一定のサイズのデータを一定の間隔で読み込むもので，1 次元の配列に対して間隔を開けたアクセスを行った場合に見られるパターンである．2 つめは飛び飛びのパターンが等間隔で現れるアクセスパターンであり，これは 2 次元の配列に対してストライドアクセスを行った場合に見られる．前もって転送する場合，どこがアクセスするかを考慮しなければ，全体に対してアクセスされる割合（密度）によって，アプリケーションから見た実効のスループットが決まる．これを向上するためには，パターンを検知する等の方法により，効率的な転送を行う必要がある．これについての提案手法は次章で述べる．ランダムアクセスは，ファイル中の任意の位置をランダムにアクセスするパターンであり，この場合先読みなどはほとんど効果が期待できない．従って，アプリケーションが前もって非同期に I/O を発行しない場合，どこがアクセスされるのかを事前に知ることは出来ず，ファイルシステム側で最適化を行うことは難しい．

従って，本論文では非同期 RPC によるシーケンシャルアクセスとストライドアクセスの最適化について述べる．

© 2011 Information Processing Society of Japan

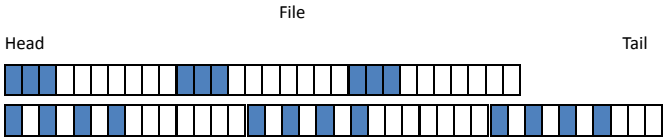

図 6 2 種類のストライドアクセス例
Fig. 6 Two types of stride access

4. 提案手法

4.1 アクセスパターンの認識

これまで，同期型 RPC を用いた遠隔ファイルアクセスの最適化に取り組む中で，アクセスパターンの認識方法を提案してきた．これは，回線帯域と遅延を考慮し，あるアクセスが連続・不連続アクセスのどちらとして捉えるべきかを導きだし，それによって RPC バッファサイズを変更するものである．本論文においては非同期型 RPC を用いるが，基本的にはこれまでに提案した手法によってアクセスパターンの判別を行う．

4.2 非同期 RPC による変更点

非同期型 RPC を用いることによって変わる点は 2 つあり，1 つはバックグラウンド転送が行われる事と，もう 1 つはストライドアクセスの際に予測転送が行えるようになることである．

バックグラウンド転送とは，回線遅延による RPC 応答待ちの間にデータを転送を行うものである．この際に転送されるデータ量は，回線遅延分の時間に転送出来る量，すなわち帯域幅遅延積分である．帯域幅遅延積とは回線遅延時間と回線速度の積のことであり，例えば Gigabit Ether(125MB/s) で遅延が 100ms の場合，12.5MB となる．シーケンシャルアクセスが行われている場合には，単に，ファイル中で連続する次の場所をバックグラウンドで転送すれば良い．ストライドアクセスの場合には，単に次の領域を転送するだけでは，そこが必要とされるとは限らず，性能が向上するかどうかが分からない．従って，ストライドアクセスがどのように行われているかを判別し，パターンを検出する必要がある．その結果に基づいて，予測転送を行う事により性能を向上させる．

4.3 ストライドアクセスのパターン検出

ストライドアクセスが行われている際にバックグラウンド転送を効率的に行うためには，それがどのようなパターンであるかを検出する必要がある．検出されたパターンに基づいて非同期にデータを転送することで，低いアクセスの密度や回線遅延による性能低下を克服することができる．

そこで，ここではストライドアクセスのパターン検出方法について述べる．**図 7** は 1 次元のストライドアクセスに対して，パターンを検出する例である．1 段目はファイルの一部分を模式的に表した図で，左側が先頭方向であり，青色の部分はアクセスされた部分を示している．この一段目はファイルの全てを示している訳では無く，直近にアクセスされた部分の一定量 (s) である．この部分をストライド検出範囲と呼ぶ事とする．この範囲のアクセスされたかどうかの情報はブロックサイズ b ごとに管理する．ブロックに属する領域が一度でもアクセスされた場合，そのブロックのフラグを立てる．2 段目はストライドアクセスのパターンを表現するための領域を示している．この領域のサイズを (d) とする．ストライド検出範囲の対応する部分から状態（アクセスの有無）をコピーし，後ろの隣接する部分と比較する．さらに，その先の隣接した部分でも状態が一致するかどうかを確認する．これを n 回繰り返して一致すれば，d とコピーされた領域を以て，ストライドアクセスのパターンとする．一致しない場合には，d を増加させ，同じ手順を行う．

上記の手順を繰り返してもストライドアクセスのパターンが得られなかった場合，それはランダムアクセスであるか，与えられた s の範囲では検知できないスケールのストライドアクセスである．

d が持つ意味は非常に大きく，1 次元以外のストライドアクセスを検知するために必要である．例えば，図 6 の 2 段目に示したような，2 次元配列に対して一部分だけアクセスした際のアクセスパターンでは，ある程度大きな領域で判断しなければ，どのようなストライドアクセスが行われているかを把握することが出来ない．もし d が小さすぎると，ストライドアクセスの一部分のみを捉えてしまう可能性がある．そのような場合には，本来必要な領域が転送されず，ミスとなる．**図 8** はこのような場合の具体例である．1 段目は同じくファイルを示しており，2 段目は d が小さすぎる場合の状態を表している．この場合はシーク幅が d と等しくなり，3 回目の場所では一致しない．これでは検知する範囲が小さすぎるので，2 段目のように d が広がると，ストライドアクセスのパターンを正しく捉えられ，d の領域を一つの周期と考えてバックグラウンド転送に用いることが出来る．

4.4 検出したパターンによるバックグラウンド転送

検知したストライドパターンに基づいて，必要であると見込まれる領域について，非同期 RPC を発行して転送を行う．その様子を表したのが**図 9** である．サイズ d の領域を基準に先回りして非同期に転送する．この領域のパターンで繰り返す可能性が高いと考えられるの

© 2011 Information Processing Society of Japan

Vol.2011-HPC-130 No.30
2011/7/28

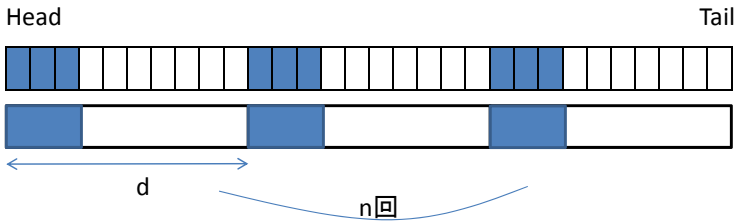

図 7 ストライドパターン検知方法
Fig. 7 A method to detect the pattern of stride access

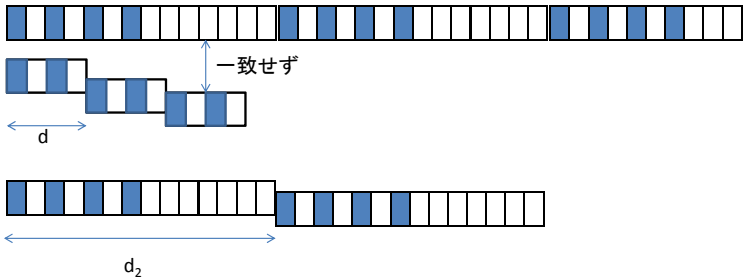

図 8 d の違いによる検出結果の違い
Fig. 8 Difference of detection result by d

で，バックグラウンドで転送されるデータ量が帯域幅遅延積を上回るまで複数回転送すると最大の性能が出せるようになる．

5. 性能評価

5.1 評価方法

本論文執筆時，非同期転送を行う評価用プログラムは実装できているが，ストライドアクセスのパターン検知等の実装が完了していない．従って，その部分についてはシミュレーションによって予測した効果と性能向上について示す．

実測値と書かれたものについては実際に性能測定を行っている．環境は，2 台のサーバを Gigabit Ethernet で接続したものであり，ネットワーク遅延は Linux の tc コマンドにより，ソフトウェアで発生させている．

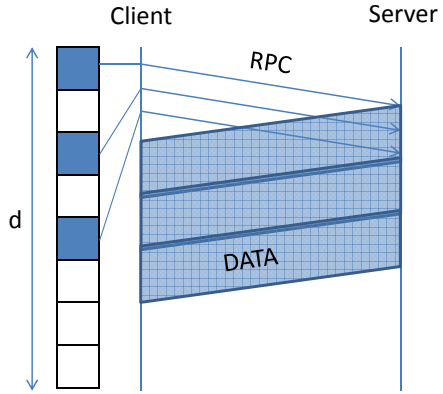

図 9 検出したパターンによる非同期転送
Fig. 9 Asynchronous data transferring by the detected access pattern

5.2 シーケンシャルアクセス

図 10 は同期型と非同期型 RPC によるシーケンシャルアクセスの性能比較（実測値）である．Blocking が同期型，Non-blocking が非同期型である．Y 軸はスループット (MB/s) で，X 軸は回線遅延である．RPC バッファサイズは双方とも 4MB に固定している．

この図から，非同期型 RPC を用いた場合には同期型と比べて，回線遅延の増大の影響をあまり受けないことが分かる．特に 25ms の回線遅延においては 1.8 倍の性能となっている．これは，同期型 RPC でデータ転送を行っている時間と応答待ち時間の比から計算した値と等しい．同期型 RPC の場合，4MB のデータ転送と 25ms 遅延時の Gigabit Ethernet の帯域幅遅延積 3.125MB から，転送している時間の割合は 56%と求められる．非同期 RPC のスループットにこの値を掛けると，ほぼ 60MB/s になり，同期型の場合の実測値とほぼ一致する．このことから，非同期に転送を行うと応答待ち時間を有効に活用し，性能を向上出来ていることが分かる．

5.3 ストライドアクセス

ストライドアクセスに関しては，本論文ではシミュレーションによる性能見込みについて述べる．このシミュレーションでは，TCP のウィンドウサイズ等の要因は考慮していない．

1 つ目の条件として，1 次元配列を格納したファイルに対し，1MB の読み込みと 3MB のシークを繰り返すストライドアクセスを想定する．以降，ブロックサイズ b は 128KB，s を

© 2011 Information Processing Society of Japan

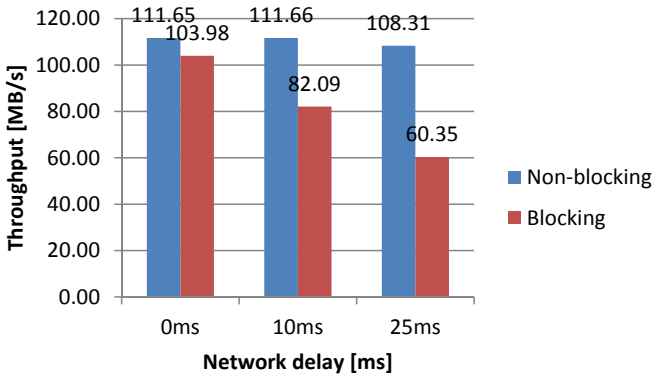

図 10 シーケンシャルアクセスの性能比較
Fig. 10 Performance comparison of sequential access

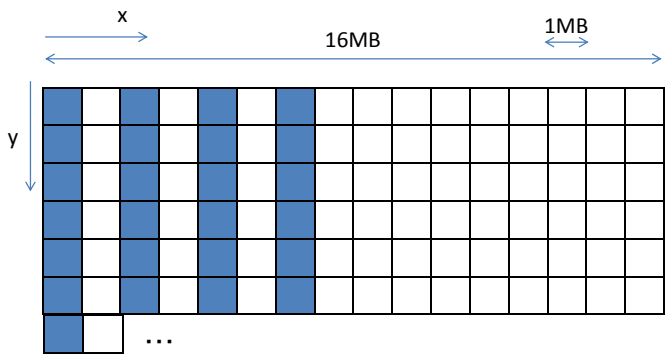

図 11 2 次元のストライドアクセス
Fig. 11 2-d stride access

128MB，n を 8 とする．この場合，まず 128MB 分，すなわち 32 回のストライドアクセスが行われる．その後，d は 128KB からスタートし，4MB まで増加した時点で条件を満たす．従って，4MB の周期のストライドアクセスであると判定される．この時点で，バックグラウンドで必要な領域を転送出来るようになり，スループットはほぼシーケンシャルアクセスと同等になる事が見込まれる．

2 つめの条件として，2 次元配列（列のサイズが 16MB）を格納したファイルに対し，ストライドアクセスを行うパターンを考える．1 列当り，先頭から 1MB 読み込みと 1MB のシークを交互に 4 回繰り返すと想定する（**図 11**）．まずは s の 128MB 分データ転送が行われる．そして，d が 2MB になった時点で一致する．しかしながら，n は 8 であるのに対して，領域の後方に対して 4 回しか一致しない．そこで，d が更に拡張される．d が 4MB になった時点でも一致が見られるが，これも後方に対して 2 回しか一致しない．d は更に増加し，16MB になった時点で後方に対しての一致回数が 8 に達する．ここで，この 16MB の単位，すなわち 2 次元上の一列がストライドアクセスの 1 周期として認識される．以後はこの単位について，必要な領域のみをバックグラウンド転送すれば良い．このように作用すれば，無駄が生じること無く連続的にデータを転送することが可能であり，シーケンシャルアクセスと同等のスループットを達成出来ると見込まれる．

6. まとめと今後の課題

本論文では，これまで行ってきた同期型 RPC による遠隔ファイルアクセスの最適化の延長として，非同期型 RPC を用いたバックグラウンド転送および，そのために必要なストライドアクセスのパターン検知の手法について述べた．そして，それらの手法が性能向上に寄与するかどうかについて，データとシミュレーションを交えて示した．

掲載したデータの一部はシミュレーションに基づくものであり，今後実アプリケーションやファイルシステムに実装することで，より実環境に即した評価を行うことは今後の課題である．また，提案手法についても，考慮が不十分なパラメータが存在している．特にパラメータ n は固定値では機能しない可能性が高く，adaptive な決め方が求められる．今後，より様々な条件において検討を進め，性能向上の妥当性を検証する必要がある．

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．

© 2011 Information Processing Society of Japan

Vol.2011-HPC-130 No.30
2011/7/28

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.: New Generation Computing, Ohmsha, Ltd. and Springer, *Gfarm Grid File System*, Vol.28, No.3, pp.257–275 (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, *In ALS 00: Proceedings of the 4th annual Linux Showcase and Conference* (2000).
5) Braam, P.J.: Lustre, *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. Computers*, 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).

© 2011 Information Processing Society of Japan