rsyncプロセスの負荷分散によるSONAS非同期コピーのパフォーマンス改善

松井 壮介　三好 浩之　高井 聡　荒木 博志

Performance Improvement of SONAS Asynchronous Replication by Load Balancing of rsync Processes

Sosuke Matsui, Hiroyuki Miyoshi, Satoshi Takai and Hiroshi Araki

IBM Scale Out Network Attached Storage（SONAS）はエンタープライズ向けのストレージ製品であり，災害対策のための非同期コピーをサポートする．2010 年 11 月に発売した R1.1.1.2 では，ファイル・システムのディレクトリー競合問題に対応し，非同期コピーのパフォーマンスを改善した［1］．しかし，R1.1.1.2 で実装された手法では，非同期コピーが並列に実行する複数のファイル転送プロセスのうち，特定のプロセスに負荷が集中し，パフォーマンス・ボトルネックとなるケースがあることが判明した．そこでわれわれは，比較的サイズの大きなファイルを転送する場合はディレクトリー競合問題の影響が少ないという点に着目した．サイズの大きなファイル群についてはサイズを考慮し，複数のファイル転送プロセスの負荷を分散させ，最大 48.3% のパフォーマンス改善効果を得た．本論文ではファイル転送プロセスの負荷分散によるパフォーマンス改善の効果を示す．本論文で提案する手法は 2012 年 5 月に発売された SONAS R1.3.1,および IBM Storwize V7000 Unified R1.3.1 に実装された．

IBM Scale Out Network Attached Storage (SONAS) supports asynchronous replication for disaster recovery. SONAS R1.1.1.2, which was released in November 2010, solved the directory collision problem and enhanced the performance of asynchronous replication [1]. However, we have found that asynchronous replication on R1.1.1.2 does not balance work-load of file replication processes under certain circumstances. As a result, a certain replication process becomes a bottleneck and degrades the performance. We focused on the fact that an overhead due to the directory collision problem can be ignored when replicating large files. We will propose a work-load enhancement that distributes large files equally to each file replication processes. Our method showed up to 48.3% performance enhancement. In this paper, we will show that our method eliminates the bottleneck and enhances the performance of asynchronous replication. The proposed method has been applied to SONAS R1.3.1 and IBM Storwize V7000 Unified R1.3.1, which were released in May 2012.

Key Words & Phrases：非同期コピー，災害復旧，SONAS，負荷分散，高速化
Asynchronous Replication, Disaster Recovery, SONAS, Load Balancing, Speeding Up

1. はじめに

非同期コピーとは，運用ストレージのデータのバックアップを，ホストからの I/O とは非同期に，遠隔地で取得するアプリケーションであり，多くのストレージ製品でサポートされている．非同期コピーのパフォーマンスを改善することで，より新しいデータをバックアップできるため，これまで多くのストレージ製品で効率の良い方法が考えられてきた［2］［3］．

IBM Scale Out Network Attached Storage（SONAS）はファイル・システムに General Parallel File System（GPFS）を採用したクラスター型のファイル・サーバーであり，2010 年 8 月に発売した R1.1.1 から非同期コピーをサポートする．SONAS の非同期コピーでも，R1.1.1 以来,パフォーマンスの改善を継続している［1］．

複数のノードが GPFS 上の同一ディレクトリー下にファイル等を新規作成すると，各ノードが同一ディレクトリーを更新することになる．このとき，GPFS の整合性を保つために，当該ディレクトリーを更新する権利をノード間で受け渡しながら，各ノードによる更新処理を順番に実行していく．このノード間での更新権の受け渡しによるオーバーヘッドにより,R1.1.1 以前の SONAS の非同期コピーではパフォーマンス劣化が問題となった［1］．

上記の問題に対処するために，2010 年 11 月に発売

提出日:2012年8月17日　再提出日:2013年2月25日

したR1.1.1.2では同一ディレクトリー下にあるファイルは同一ノードが転送することで，パフォーマンスを改善した［1］．しかし，R1.1.1.2の発売後，あるディレクトリー下にサイズの大きいファイルが集中する場合，特定のノードに負荷が集中し，非同期コピーのパフォーマンス・ボトルネックとなるという問題が発生した．1つのディレクトリー下に同種のファイルを保存するファイル・ツリー構成は一般的であり，仮想マシンのイメージ等のサイズの大きいファイルがあるディレクトリー下に集中する環境は少なくない．上記の問題が非同期コピーをお使いの全てのお客様環境で発生する可能性があり，早急に対策を講じる必要があった．

そこで，われわれは，各ノードのファイル転送処理の負荷を分散し，かつディレクトリーの更新権の受け渡しによるオーバーヘッドを解消するパフォーマンス改善手法を提案する．本手法を適用することで，SONASシミュレーター環境で最大48.3%パフォーマンスを改善するという効果が得られた．

2. SONAS非同期コピーのパフォーマンス問題

2.1 SONASとは

Network Attached Storage（NAS）とは，ストレージ装置にファイル・システムを内蔵したファイル・サーバー専用装置であり，ユーザーはイーサネット経由でNAS上のデータにファイル単位でアクセスする．SONASとは，ファイル・システムにGPFSを採用したクラスター型のNAS製品であり，複数のノードから構成され，各ノードが同一ファイル・システムにアクセスする．ファイルを複数のノードに分散し，並列にI/Oを実行することで，高速なファイル・アクセスを実現する［4］．

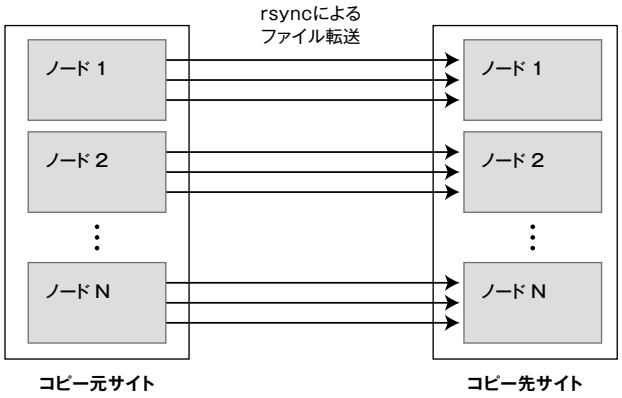


図1. SONAS非同期コピーによるファイル転送の概略図

2.2 SONASの非同期コピーとは

SONASの非同期コピーは，2つのサイト間でデータ転送を行い，コピー元サイトのファイル・ツリーの複製を，コピー先サイトにて取得する．非同期コピーはファイル単位で行われ，rsync［5］を用いることでコピー元サイトからコピー先サイトへ差分データのみネットワーク転送される．**図1**のように，コピー元サイトとコピー先サイトのノードは1対1の関係になっており，各ノードで複数のrsyncプロセスが並列に実行される．非同期コピーに使用するノードの数（最大30台）と，各ノードで実行するrsyncプロセスの数（最大10プロセス）は管理者が自由に設定できる．

2.3 GPFSのディレクトリー競合問題

あるノードがGPFSにファイル等を新規作成する場合，まず，そのファイルの親ディレクトリーを更新する権利を取得し，次いでそのディレクトリーに対する更新処理を実行する．ディレクトリーの更新処理とは，新規作成するファイルを親ディレクトリーに追加する処理を指す．複数のノードが同一ディレクトリーに同時にファイル等を新規作成する場合，ディレクトリーを更新する権利をノード間で受け渡し，順に更新処理を実行する．このとき，ノード間での更新権の受け渡し処理によるオーバーヘッドが発生してしまう（以降，ディレクトリー競合と呼ぶ）．

SONASの非同期コピーでは複数のノードが並列にrsyncを用いてファイル転送をするため，異なるノードで実行されるrsyncプロセスがコピー先サイトの同一ディレクトリー下にファイルを転送する場合，ディレクトリー競合が発生する．R1.1.1.1以前の非同期コピーでは，このディレクトリー競合によるパフォーマンスの劣化が問題となった．

ディレクトリー競合による非同期コピーのパフォーマンスの劣化の度合いを確認するために，テスト環境でパフォーマンス検証を行った．検証のために，Linuxの Kernel-based Virtual Machine（KVM）を用いたSONASシミュレーターを2つ用意し，シミュレーター間でファイル転送に要する時間を測定した．各SONASシミュレーターに3つずつ，合計6つのノードを用意した．検証の手順は以下の通りである．

① SONASシミュレーターのGPFSにディレクトリーを3つ作成する．
② 各ディレクトリー下に10KBのファイルを3,000個作成する．
③ 3つのノードにrsyncプロセスを1つずつ並列に実行させ，②で作成したファイルを別のSONASシミュレーターに転送する．

④ ファイル転送を開始してから終了するまでの時間を測定する.

上記の手順で, 以下の 2 つの転送方法による実行時間の比較を行った.

- 各ディレクトリーのファイルを 1,000 個ずつ, 3 グループに分割する. 各ノードで実行される rsync プロセスは, 同時に同一ディレクトリー内の異なるグループのファイルを転送する（ディレクトリー競合が発生する）
- 各ディレクトリーのファイルは, それぞれ異なるノードで実行される rsync プロセスが転送する（ディレクトリー競合は発生しない）

測定結果を**表 1** に示す. この結果から,転送するデータ量は等しいにもかかわらず, ディレクトリー競合が発生する場合は発生しない場合よりも遅くなり, オーバーヘッドの大きさが無視できないことを再確認できた.

表 1. ディレクトリー競合がファイル転送のパフォーマンスに与える影響

	ファイル転送に要する時間（秒）
ディレクトリー競合が発生する場合	146
ディレクトリー競合が発生しない場合	41

2.4 SONASの非同期コピー・サービス

R1.1.1.2 における非同期コピーのフローは**図 2** のようになっている.

① コピー元サイトにおいて, 前回バックアップ作成時から現在までに更新があったファイルのリストを, GPFS の高速スキャン機能を用いて作成する. ファイル・リストには, ファイルの絶対パスが記載される.
② 作成されたファイル・リストを, 絶対パスでソートする.
③ 絶対パスでソートされたファイル・リストを, 起動する rsync プロセスの数で分割する（**図 2** では 3 つ）. 各分割ファイル・リストに記載されるファイルの数は均等にする.
④ 各 rsync プロセスが分割ファイル･リスト上のファイルを, コピー先サイトに並列に転送する.

上記の②においてファイル・リストを絶対パスでソートすることで, 同一ディレクトリー下に存在するファイルがファイル･リストの中で連続して現れる. 従って, 同一ディレクトリー下にあるファイルが同一分割ファイル・リストに連続して記載される可能性が高くなる. 各 rsync プロセスはおのおのが担当する分割ファイル・リストに記載されたファイルを, リストの上から順に転送する. これにより, 異なるノードがコピー先サイトの同一ディレクトリー下にファイルを転送する可能性を低くし, コピー先サイトでディレクトリー競合の発生を回避することができる. R1.1.1.2 では非同期コピーにこの手法を適用することで, 最大 48%のパフォーマンス改善に成功した［1］.

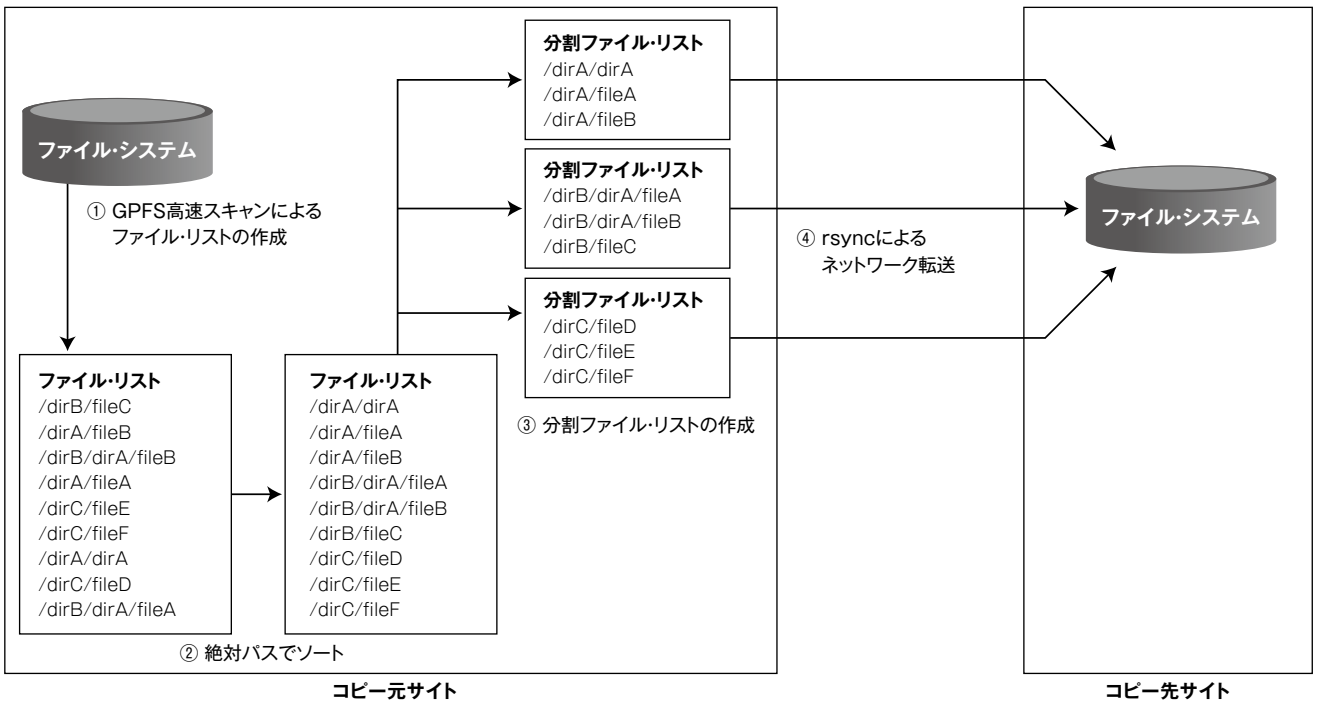


図2. SONAS非同期コピーの流れ

2.5 rsyncプロセスの負荷集中問題

上記の対策を講じることで，ディレクトリー競合によるパフォーマンス劣化の解消に成功した．しかし，R1.1.1.2の発売後，ファイル・サイズの大きい仮想マシンのイメージ･ファイルやデータベース･ファイル等が同じディレクトリーに集中する場合，新たな問題が発生することが明らかになった.2.4で述べた方法では絶対パスでソートされたファイル・リストを，ファイルの数が均等になるように分割するため，サイズの大きいファイルが1つのディレクトリーに集中すると，ある分割ファイル・リストに記載されるファイル・サイズの合計値が他の分割ファイル・リストと比較して，突出して大きくなることがあった．このため，ファイル･ツリーの構成によって，非同期コピーを実行するとサイズの大きなファイルが特定のrsyncプロセスに集中し，パフォーマンス・ボトルネックとなる問題が発生した．

この問題を調査するために，SONASシミュレーター環境の特定のディレクトリー下に比較的サイズの大きいファイルを用意し，以下の手順で問題を再現した．

① あるディレクトリーに1GBのファイルを10個作成する．
② 別のディレクトリー下に各サブディレクトリーにファイルが6個,ディレクトリーが5個存在する深さが4のファイル･ツリーを作成する．各ファイルのサイズは1MBとする．
③ ノードを3つ用いて非同期コピーを実行し，各rsyncプロセスの実行時間を計測する．各ノードではrsyncプロセスを1つずつ実行する．

各rsyncプロセスの実行時間は**表２**のようになった．ノード1で実行されたrsyncプロセス（rsyncプロセス1）が1GBのファイル10個を転送したため，そのプロセスの実行には748秒必要であった．一方，その他のノードで実行されたrsyncプロセス（rsyncプロセス2，3）の実行時間はそれぞれ91秒，86秒であった．以上により，特定のrsyncプロセスがパフォーマンス・ボトルネックとなる問題を再現できた．

表 2. 提案手法適用前の，各 rsync プロセスの実行時間

	実行時間（秒）
ノード1（rsyncプロセス1）	748
ノード2（rsyncプロセス2）	91
ノード3（rsyncプロセス3）	86

3. 提案手法

非同期コピーのパフォーマンス・ボトルネックを解消するために，われわれは以下の二つの問題を同時に解決する必要があった．

- 特定のrsyncプロセスにサイズの大きなファイル群の転送処理が集中するために発生する,パフォーマンス･ボトルネック
- 複数のノードがコピー先サイトの同一ディレクトリー下にファイル転送をする際に発生する，ディレクトリー競合によるパフォーマンス劣化

上記の2つの問題を同時に解決するために，われわれは，ディレクトリー競合がコピー先サイトでディレクトリーを更新する際に発生するという点に着目した．比較的サイズの大きいファイルをネットワーク転送した場合，コピー先サイトではディレクトリーの更新処理に対してファイル・データの書き込み処理が多く発生することが予想される．すなわち，複数のノードがサイズの大きなファイルをコピー先サイトの同一ディレクトリー下に転送した場合，ディレクトリー競合によるオーバーヘッドに対してデータ転送に要する時間が十分に大きく，オーバーヘッドは無視できると考えられる．

そこで，われわれの提案手法では，サイズの大きいファイル群については負荷が分散されるように，分割ファイル・リストを作成する．一方，サイズの小さいファイル群については，R1.1.1.2で実装された手法を適用し，同一ディレクトリー下のファイルを同一分割ファイル・リストに記載する．

提案手法による非同期コピーの処理の流れは**図３**のようになる．

① コピー元サイトにおいて,GPFS高速スキャンをかけて，ファイル・リストを作成する．このとき，ファイルの絶対パスとファイル・サイズが表示される．
② 作成されたファイル・リストを一行ずつ読み，サイズがある閾値 T 以上のものをラージ・ファイル・リスト，閾値 T 以下のものをスモール･ファイル･リストに記載する．この閾値は外部パラメータから調整可能である．
③ ラージ・ファイル・リストに記載されたファイルを上から順に，各rsyncプロセスに配るようにして分割ラージ・ファイル・リストに記載する．分割ラージ・ファイル・リストに記載されたファイル・サイズの合計がある閾値以上になった場合，新しい分割ラージ・ファイル・リストを作成し，そこに記載する．分割ラージ・ファイル・リストのサイズに上限を設けることで，各リスト間のサイズのばらつきを制限することができる．（**図３**の場合は30MB，5GB，および1GBのファイルの転送を異なるrsyncプロセスが担当する.SONASの非同期コピー

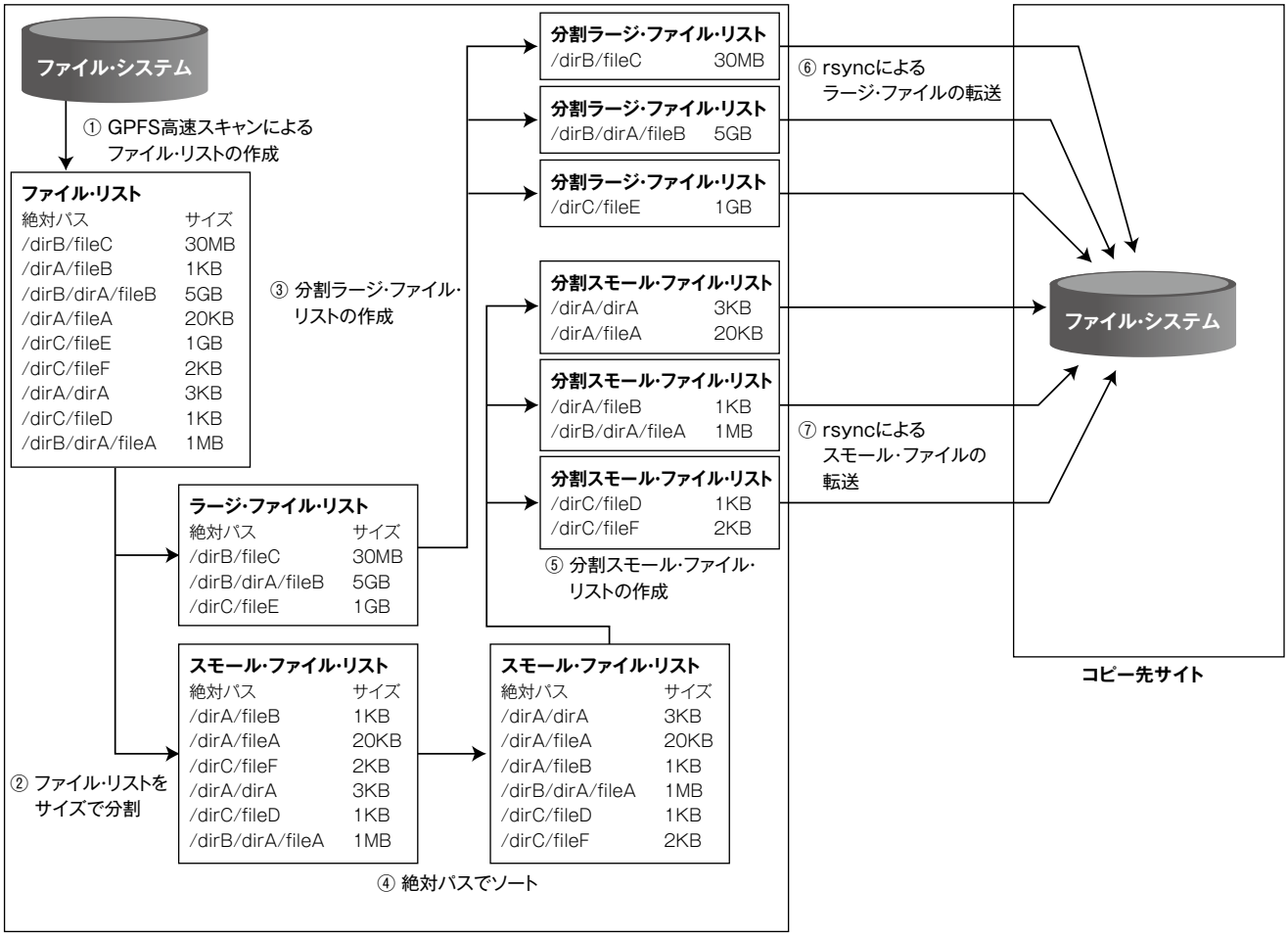


図3. 提案手法の流れ

はファイル単位であるため，ファイルを分割して rsync プロセス間で負荷を分散することはできない.）

④ スモール・ファイル・リストを絶対パスでソートする.

⑤ ソートされたスモール・ファイル・リストを，起動する rsync プロセスの数で分割する.

⑥ 各 rsync プロセスが，分割ラージ・ファイル・リストに記載されたファイルを順にネットワーク転送する.

⑦ 各 rsync プロセスが，分割スモール・ファイル・リストに記載されたファイルを順にネットワーク転送する.

④で分割ラージ・ファイル・リストのサイズに上限を設けることで，rsync プロセスの負荷を制限することができる. また，分割スモール・ファイル・リストには，同一ディレクトリー下のファイル群が記載される. 以上により，サイズの大きいファイルについては負荷分散が，サイズの小さいファイルについてはディレクトリー競合の解消が実現される.

4. 検証

提案手法によるパフォーマンス・ボトルネックの解消，およびパフォーマンス改善の効果を見るために，まず提案手法の着眼点である仮説が正しいことを検証し，次いで提案手法によるパフォーマンス改善の効果を測定した. 最後に，提案手法による分割ファイル・リストの生成方法と以前の方法との比較を行った.

4.1 ファイル・サイズとディレクトリー競合の関係

ファイル・サイズを変えて複数ノードからコピー先サイトの同一ディレクトリー下にファイル転送を行い，ディレクトリー競合によるオーバーヘッドがファイル転送に要する時間に占める割合を計測した. まず，GPFS にディレクトリーを 3 つ用意し，各ディレクトリーにファイルを 810 個作成した. 次に，ノードを 3 つ用いて，各ノードに異なるディレクトリー内のファイルを並列に転送させた場合の時間（以下，競合なし時間 t_n とする.）と，各ディレクトリー内のファイルを 270 個ずつのグループに分け，各ノードに同一ディ

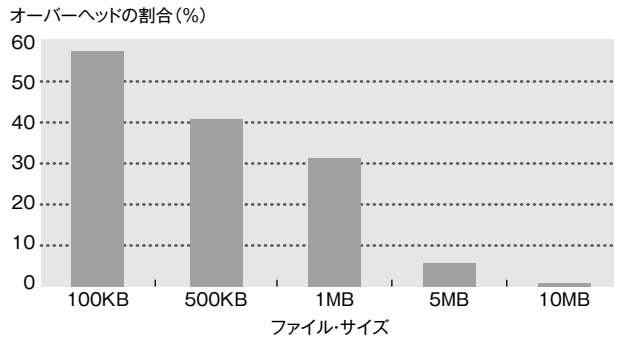


図4. ファイル転送時間に占めるディレクトリー競合による オーバーヘッドの割合

レクトリー内の異なるグループのファイルを並列に転送させた場合の時間（以下，競合あり時間 t_c とする.）を計測した．ディレクトリー競合によるオーバーヘッドを $t_o=t_c-t_n$，ファイル転送に要する時間に占めるオーバーヘッドの割合を $P_o=t_o / t_c \times 100$ と定義すると，ファイル・サイズを 100KB，500KB，1MB，5MB，10MB と変えた場合，P_o は図 4 のように変化した．

ファイル・サイズが大きくなるほど，転送時間に占めるオーバーヘッドの割合は減少し，10MB のファイルを転送した場合に 0.53% となった．すなわち，比較的大きなサイズのファイルを転送する場合，ディレクトリー競合によるパフォーマンスの劣化はほぼない．ディレクトリー競合によるオーバーヘッドが無視できるのであれば，2.4 の表 2 のようなケースでは，ディレクトリー競合を回避するために 1GB のファイル群を同一プロセスで転送するよりも，複数プロセスで並列に転送したほうが，効率が良いと予想できる．

4.2 提案手法の効果

次いで，提案手法によるパフォーマンス改善の効果を測定した．まず，サイズの大きいファイルが同一ディレクトリーに集中する，という環境を再現するために，同一ディレクトリーに 1GB のファイルを 10 個作成した．次に，同一ディレクトリーに集中しないファイル・ツリーを生成するために，1KB ～ 1MB のサイズのファイルを A%，1MB ～ 10MB のファイルを B%，10MB ～ 100MB のファイルを C%，100MB ～ 1GB のファイルを D%，1GB のファイルを E% の確率で作成するプログラム *P(A,B,C,D,E)* を用意した．このプログラムを用いて，各サブディレクトリーにファイルが 3 つ，ディレクトリーが 4 つ存在する深さが 4 のファイル・ツリーを作成した．ファイル・サイズの上限を 1GB としたのは，SONAS シミュレーターの容量の制限のためである．A ～ E の値を変えて提案手法の適用前と後のファイル転送処理のスループットを比較すると，図 5 のような結果が得られた．いずれのファイル・ツリー構成でも，提案手法の適用により同一ディレクトリーに保存された 1GB のファイル群の処理がノード間で分散されたため，非同期コピーのパフォーマンスが改善し，最大 48.3% の改善効果が得られた．なお，実環境に近い環境で測定するため，非同期コピーにはノードを 5 つ用いた．各ノードでは rsync プロセスを一つずつ実行した．分割ファイル・リストの生成方法を変えるファイル・サイズの閾値 T は 10MB に設定した．

また，パフォーマンス改善の効果が最大となった $P(75,10,10,5.0)$ の場合に，各ノードの rsync プロセスの実行時間を計測したところ，図 6 のようになった．提案

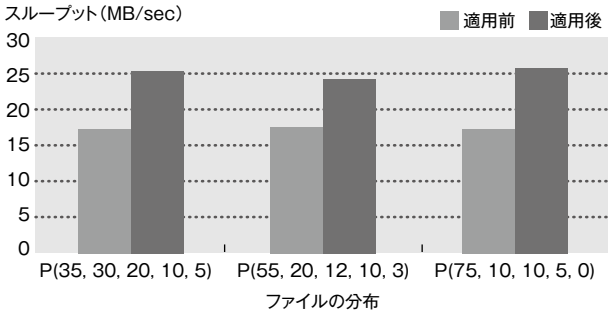


図5. 提案手法適用前と適用後の非同期コピーによる ファイル転送処理のスループット比較

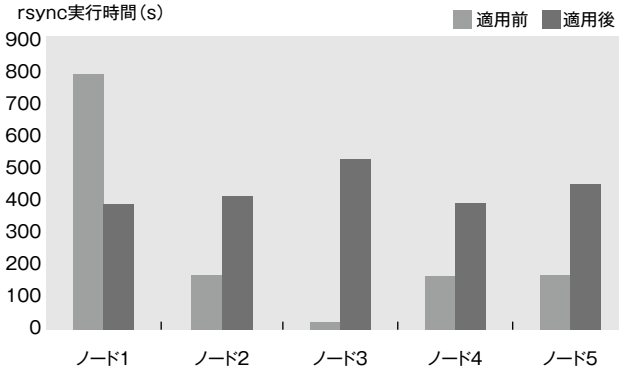


図6. 各rsyncプロセスの実行時間

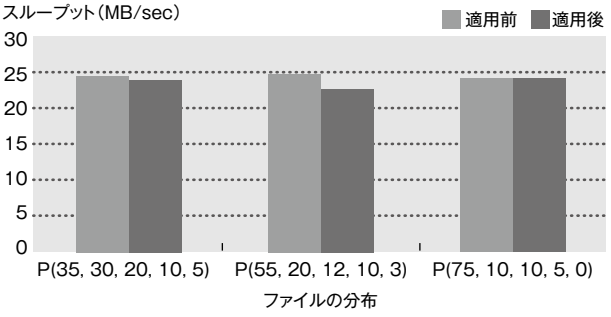


図7. 提案手法適用前と適用後のファイル転送処理のスループットの比較 （サイズの大きいファイルが同一ディレクトリーに集中していない場合）

手法を適用することで，ノード 1 に集中していた負荷を分散した．

4.3 同一ディレクトリーにサイズの大きなファイルが集中しない場合

提案手法による改善が期待できないケースを明らかにするために，4.2 の実験環境から，1GB のファイルを 10 個含んだディレクトリーを削除して，提案手法適用前と適用後のパフォーマンスを測定した．結果は図 7 のようになった．提案手法適用前においてパフォーマンス・ボトルネックが発生しなかったため，適用後に改善の効果が見られなかった．しかし，提案手法の適用によってパフォーマンスが大きく劣化してしまうことがほぼないことも確認できた．

5. 結論

本論文では，サイズの大きいファイルは GPFS のディレクトリー競合の影響が少ない，という点に着眼し，rsync の負荷分散手法を提案した．SONAS シミュレーターによる実験から，提案手法はこれまでの手法と比較して，最大 48.3%のパフォーマンス改善効果が得られた．あるファイル・ツリー構成では提案手法による改善の効果は見られなかったが，多くのお客様環境では同一ディレクトリーにサイズの大きいファイルが集中するファイル・ツリー構成であるため，提案手法による改善の効果が期待できる．

本論文で提案した手法は，2012 年 5 月に発売した SONAS R1.3.1，および SONAS と共通のソフトウェア・スタックを持つ Storwize V7000 Unified R1.3.1 で実装され，より高性能な転送機能を提供できた．

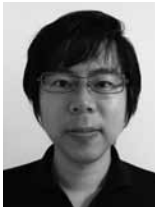
参考文献

［1］ 三好 浩之，萩原 克彦，松井 壮介，岩崎 礼江：SONAS 非同期コピーのパフォーマンス改善，PROVISION No.70，http://www.ibm.com/ibm/jp/provision/no70/pdf/70_paper2.pdf
［2］ Cronauer, P., Dufrasne, B., Altmannsberger, T. et al.: IBM System Storage DS8000 Copy Services for Open Systems, IBM Redbooks, http://www.redbooks.ibm.com/abstracts/sg246788.html
［3］ EMC: Best Practices For Data Replication with EMC Isilon SyncIQ, http://www.emc.com/collateral/hardware/white-papers/h8224-replication-isilon-synciq-wp.pdf
［4］ Lovelace, M., Boucher, V., Nayak, S.et al.: IBM Scale Out Network Attached Storage: Architecture, Planning, and Implementation Basics, IBM Redbooks, http://www.redbooks.ibm.com/abstracts/sg247875.html
［5］ rsync, http://rsync.samba.org/

日本アイ・ビー・エム株式会社
システム・テクノロジー開発製造．
ストレージ・システムズ開発
ソフトウェア・エンジニア

松井 壮介 Sosuke Matsui

［プロフィール］
2009 年，日本 IBM 入社．IBM 東京ラボラトリーにて，SONAS のコピーサービスの設計・開発に従事．情報処理学会会員．
e34975@jp.ibm.com



日本アイ・ビー・エム株式会社
システム・テクノロジー開発製造．
ストレージ・システムズ開発
スタッフ R&D エンジニア

三好 浩之 Hiroyuki Miyoshi

［プロフィール］
2001 年，日本 IBM 入社．ミッドレンジからエンタープライズのストレージ製品のマイクロコードの設計・開発に従事．現在は SONAS のコピーサービスの設計・開発を担当．

日本アイ・ビー・エム株式会社
システム・テクノロジー開発製造．
ストレージ・システムズ開発
ソフトウェア・エンジニア

荒木 博志 Hiroshi Araki

［プロフィール］
2008 年，日本 IBM 入社．エンタープライズのストレージ製品の管理・監視アプリケーションの設計・開発に従事．現在は SONAS のコピーサービスの設計・開発を担当．

日本アイ・ビー・エム株式会社
システム・テクノロジー開発製造．
ストレージ・システムズ開発
スタッフ R&D エンジニア

高井 聡 Satoshi Takai

［プロフィール］
2002 年，日本 IBM 入社．ソフトウェア製品の開発．テープドライブのマイクロコードの設計・開発に従事．現在は SONAS のコピーサービスの設計・開発を担当．