SONAS 非同期コピーのパフォーマンス改善

三好 浩之　萩原 克彦　松井 壮介　岩崎 礼江

Performance Improvement of SONAS Asynchronous Replication

Hiroyuki Miyoshi, Katsuhiko Hagiwara, Sosuke Matsui and Norie Iwasaki

非同期コピーは,災害復旧ソリューションを実現するための機能の1つとして多くのエンタープライズ・ストレージ製品にサポートされている. IBM が 2010 年に発売した Scale Out Network Attached Storage（SONAS）の非同期コピーのパフォーマンスの検証を行ったところ, ノードの並列化処理によるパフォーマンスの向上が見られず,むしろ大幅に劣化してしまうケースがあることが判明した. そこで,パフォーマンス劣化の原因が,ファイル・システムの排他制御によるものであることを究明し, SONAS の非同期コピー・モジュールに並列処理を効率化するための対策を講じた. その結果, 並列処理による非同期コピーの完了に要する時間を最大で48% 短縮させたので報告する.

Asynchronous replication is supported by many enterprise storage products as a part of a disaster recovery solution. Scale Out Network Attached Storage (SONAS), which has been released this year by IBM, also supports asynchronous replication. We have measured the performance of SONAS' asynchronous replication and have discovered that there are cases in which parallel replications using multiple nodes cause a drop instead of an improvement in performance. We have found that the root cause of the drop in performance is due to exclusive control by the file system, and therefore added new logic to the SONAS asynchronous replication module to improve the efficiency of parallel replications. Our performance measurements prove that we have shortened the time required to complete asynchronous replication by up to 48%.

Key Words & Phrases ：非同期コピー, 災害復旧, SONAS, パフォーマンス, 並列処理
asynchronous replication, disaster recovery, SONAS, performance, parallel replications

1. はじめに

災害発生時におけるデータ保護および継続的なデータ・アクセスの確保は, 多くの企業にとって重要な課題となっている. IBM DS8000, IBM N シリーズ, 日立のハイエンドストレージといった多くのエンタープライズ・ストレージ製品は, 災害復旧ソリューションに必要となる機能として非同期コピーを提供する［1］［2］［3］. 非同期コピーとは, ホスト I/O とは非同期に, コピー先ストレージにバックアップを作成する機能である. この機能は,ホスト I/O のパフォーマンスに極力影響を与えずに, 復旧用バックアップ・データの作成を可能とする.

IBM が 2010 年 3 月に発売したエンタープライズ・ストレージ製品である Scale Out Network Attached Storage（SONAS）［4］は, IBM のストレージ・クラウドを実現するコア・テクノロジーとして注目されている. この製品は, 最大 14.4 ペタバイトをサポートする大規模 Network Attached Storage（NAS）で, ノード追加による容量およびパフォーマンスの拡張性を最大の特徴とする. そして, 災害復旧ソリューションに必要となる非同期コピーをサポートする.

今回, SONAS の非同期コピーのパフォーマンスを検証した. すると, コピー対象のファイルの大きさやディレクトリー構成によって, 複数ノードによる並列処理のパフォーマンスが, 単体ノードによる処理のパフォーマンスに比べ, 大幅に劣化してしまうことを発見した.

非同期コピーのパフォーマンスが低い場合の問題として, 以下の 2 つが挙げられる.

1 つ目は, RPO（Recovery Point Objective）［5］の値が大きくなってしまう問題である. RPO とは, 最新データとバックアップ・データの差を表すもので, この値が小さいほど, バックアップ・データが常に最新に近い状態にあることを意味する. この RPO の値が大きいということは, 最新のバックアップ・データを維持することができず, より多くのデータが障害発生時に失われてしまう可能性があることを意味する.

提出日:2010年9月6日　再提出日:2011年6月21日

2つ目は，非同期コピー・プロセスがリソースを長時間占有してしまう問題である．結果として，リソースを必要とする他のプロセスのパフォーマンスに影響する可能性がある．

そこで，SONASの非同期コピーの並列化におけるパフォーマンス劣化の原因をファイル・システム・レベルで究明した．さらにSONAS非同期コピー・モジュールに対策を講じることで，並列処理のパフォーマンスを向上させたので報告する．

2. SONASの概要

SONASの概要とその特徴を示す．

2.1 内部構成

図1に示すように，SONASはマネージメント・ノード（最大2台），インターフェース・ノード（最大30台，以降Int.ノードと記述），ストレージ・ポッド（ストレージ・ノードとストレージ・サブシステムの組み合わせ．最大30台）から構成される．内部ファイル・システムとして使用するGeneral Parallel File System（GPFS）は，大規模クラスターに対応する分散共有ファイル・システムであり，以下の特徴を持つ．

- 1つのファイル・データを複数のストレージに分散配置する．複数ストレージに並列アクセスすることで高速IOを実現する．
- 複数ノードによるデータ共有を可能にする．

SONASのユーザー・データはGPFSで管理され，全てのInt.ノードからアクセス可能である．

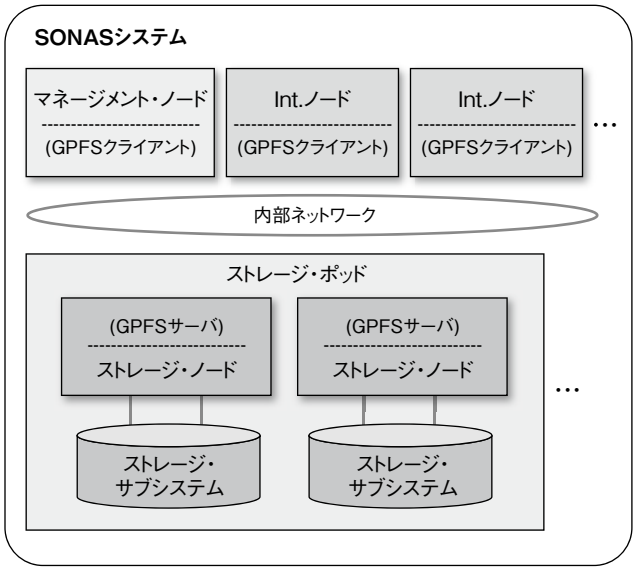


図1. SONAS内部構成図

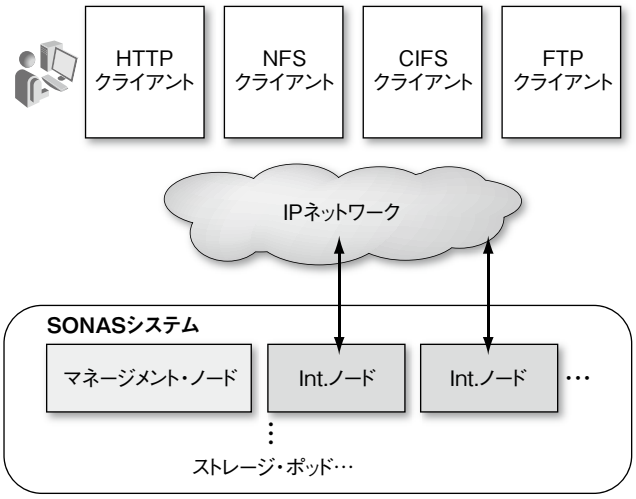


図2. SONAS外部インターフェース

2.2 外部インターフェース

図2に示すようにSONASの外部インターフェースとして，Int.ノードは，CIFS（Common Internet File System），NFS（Network File System），FTP（File Transfer Protocol），HTTP（Hypertext Transfer Protocol）をサポートする．ユーザーは外部クライント・マシンから，これらのプロトコルで，SONASのInt.ノードにアクセスを行う．

3. SONASの非同期コピー

SONASの非同期コピーは，運用SONAS（以下，ソースSONAS）から，遠隔地にあるバックアップSONAS（以下，ターゲットSONAS）にデータコピーを行う．コピーはファイル単位で行われ，rsyncを用いることで，ソースSONASとターゲットSONAS間の差分データのみをネットワークに流すようにしている．

rsyncは，2つのファイルやディレクトリーの同期をとるために，現在広く使われているUNIXアプリケーションである．同期の際，データ転送量を最小にするために，符号計算を用いて差分データを抽出し，差分データの転送を行う．

SONASの非同期コピーは，最大60,000,000ファイルのコピーをサポートする．rsyncはInt.ノードにより実行され，複数のInt.ノードを並列に参加させることで，パフォーマンスを向上させることが可能である．以下に非同期コピーのフローを示す．

1）ソースSONASにおいてコピー対象のGPFSファイル・システムのスナップショットを作成する．
2）ソースSONASのスナップショットの中で，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.

3）Int. ノードを複数台用いる場合は，ファイル・リストを分割して各 Int. ノードに渡す.

4）それぞれの Int. ノードが,自分が担当する分割ファイル・リスト上のファイルを，ターゲット SONAS に rsync で差分コピーしていく.

5）すべての Int. ノードが分割ファイル・リストの rsync 処理を完了させた後，ターゲット SONAS においてスナップショットを作成する（これがリカバリー・ポイントとなる）.

参加させる Int. ノードの数は設定可能である．上記の一連の処理を定期的に実行することでターゲット SONAS にバックアップを維持する．また，SONAS 非同期コピーの場合，一連の処理が完了した後に，次の一連の処理を実行することが可能になる．従って RPO の最小値は，この一連の処理にかかる時間に等しい.

4. SONAS の非同期コピーのパフォーマンス

SONAS の非同期コピーのパフォーマンス測定を行い，特に，参加させる Int. ノードの数を増やした場合に期待通りにパフォーマンスが向上するか検証を行った.

4.1 パフォーマンス測定方法

以下に測定方法を示す.

1）参加 Int. ノードの数を決定する.

2）ソース SONAS に，テスト・ディレクトリーおよびテスト・ファイルを作成する．ファイルの大きさと数およびディレクトリー構成はテスト・ケースによって変化させる.

3）非同期コピーを実行し，テスト・ディレクトリーおよびその下のすべてのテスト・ファイルをターゲット SONAS にコピーする.

4）ソース SONAS のテスト・ファイルのうち，n% のファイルに対し，ファイルの最後に数バイトのデータを追加する.

5）テスト・ディレクトリーおよびその下の全てのテスト・ファイルに対して非同期コピーを実行し，ターゲット SONAS への差分コピーが完了するまでの時間を計測する.

6）ステップ 4）の n の値を 0 から 100 の範囲で数点とり，その都度，ステップ 4）と 5）を繰り返す.

4.2 ベーシック・テストの測定結果

図 3 に，2KB の比較的小さいサイズの 10,000 個の比較的数多いテスト・ファイルを，一つのテスト・ディレクトリーに作成したケース（以下，ベーシック・テスト）の測定結果を示す．なお Int. ノードは 1 つしか用いていない．縦軸にコピー実行にかかった時間，横軸に更新ファイルの数を示す．縦軸の値が低いほど，パフォーマンスが良いことを示している.図 3 より,更新ファイルの数が増えると，コピー完了にかかる時間もほぼ線形に増えていることが分かる.

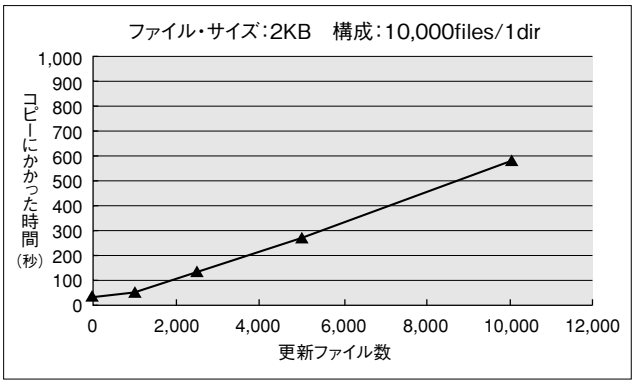


図 3. ベーシック・テスト測定結果

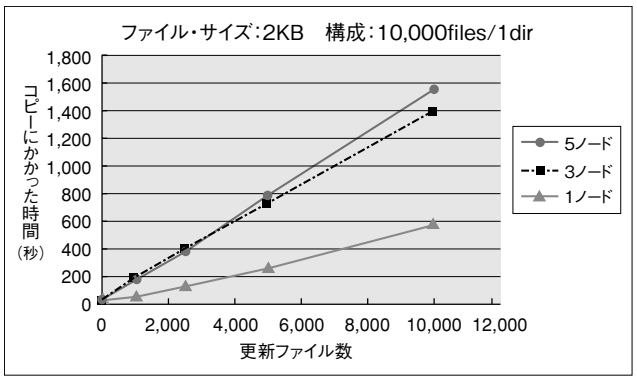


図 4. ノード数テスト測定結果

なお本測定は Linux の Kernel-based Virtual Machine（KVM）を使った SONAS エミュレーション環境で行った.

4.3 Int. ノード数の比較

ベーシック・テストと同条件にて，参加 Int. ノードの数を変化させた場合の結果を図 4 に示す（以下，ノード数テスト）.

当初の期待とは異なり，Int. ノードの数を 1 から 3，そして 5 に増やすと，コピー完了にかかる時間が増え，パフォーマンスが大幅に劣化しているのが分かる．この傾向は，図 4 に示すような，比較的小さいファイルが多く存在する場合に顕著に現れた．また割愛するが，SONAS 実機でも同測定を行い，同様の傾向を確認している．これにより,参加させる Int. ノードを増やすと,かえってパフォーマンスが劣化してしまうという問題が明らかになった.

5. 並列処理のパフォーマンス劣化の原因

並列処理によるパフォーマンス劣化の原因を突き止めるために，第 3 章で述べたフローのステップ 4）に着目した. 具体的には,各 Int. ノード上で実行される rsync が,ソース SONAS の GPFS 上のファイルにアクセスし，ターゲット SONAS の GPFS に差分コピーを完了させるのにかかる時間を調べた. I/O を複数のノードに分散し並列にアクセスする方が，1 つのノードからすべてのファイルにシーケンシャルにアクセスするよりも，実行時間が短くなることを予測していた. ところが，同一ディレクトリーに比較的小さいファイルが数多く含まれるケースにおいて，逆の傾向が見られた. 複数ノードから rsync を同時に実行すると, かえってパフォーマンスが劣化していた.

調査結果を GPFS 開発チームに確認したところ，以下の回答が得られた. あるノードがファイルなどを新規作成する際に，そのファイルの直上ディレクトリーのトークンを取得する必要がある. 従って,複数ノードが同一ディレクトリーに同時にファイルなどの新規作成を行うと，ディレクトリー・トークンの受け渡しのオーバーヘッドが発生してしまう（以下，ディレクトリー競合）.

rsync はソース SONAS とターゲット SONAS のファイルを比較し，差分を抽出･転送し，ソース SONAS とターゲット SONAS のファイルの同期を行う. この際，ターゲット SONAS 側のファイルは，新しいファイルとして作り直される. つまり，ターゲット SONAS 側では毎回ファイルの新規作成が行われる. 従って，ソース SONAS に小さいファイルが数多く同一ディレクトリーに存在する場合，ターゲット SONAS 側で，同一ディレクトリーに，ファイルの新規作成が頻繁に行われることになる. 複数ノードから実行した場合，ディレクトリー・トークンの受け渡しのオーバーヘッドが全体のパフォーマンスの中で大きな割合を占めてしまう.

問題が見られたノード数テストでは，テスト･ディレクトリーを 1 つしか用意せず，その直下に全てのテスト・ファイルを作成した. 従って，複数 Int. ノードが rsync する際，必ずディレクトリー・トークンの競合が発生していた.

試しに，テスト・ディレクトリーを数多く用意し，1 ディレクトリーに最大でも 16 ファイルしか無いようなテスト・ファイル構成を作成し，テストを実行した（以下，マルチ・ディレクトリーテスト）. この際，ファイル・サイズはベーシック・テスト同様に比較的小さいサイズ（1KB）で , トータルのファイル数は多い（約 360,000）.

測定結果（図 5）より，このテスト・ファイル構成では，Int. ノードを増やした場合，当初の期待通り，より短時間でコピーが完了し，パフォーマンスが向上することが分かる. これは，ディレクトリー競合の確率が低いためと考えられる.

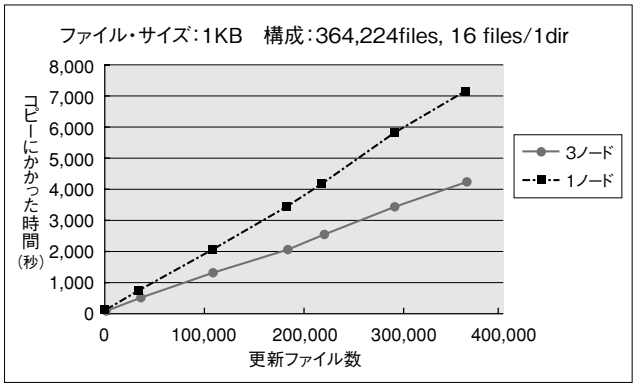


図 5. マルチ・ディレクトリーテスト測定結果

従って，Int. ノード追加によるパフォーマンス劣化問題は，少ないディレクトリーに更新されたファイルが数多く存在するケースにおいて起こる. ユーザーの実環境を考慮すると，このようなケースは十分にあり得る. その場合にパフォーマンスが向上しないどころか，かえって劣化してしまうのは大きな問題である.

このディレクトリー競合は, 現在 GPFS 開発チームによって改善中であるが，完全にオーバーヘッドをなくすことは難しく，また，高い拡張性を強みとする SONAS には極力早いタイミングで修正を入れる必要がある. そこで今回，SONAS の非同期コピー・モジュールに対策を入れ，問題を回避する方法を提案した.

6. SONAS 非同期コピーの対応策

各 Int. ノードが並列に rsync を実行する際に, 同一ディレクトリーに含まれるファイルの処理を，複数のノードが同時に実行しないようにすることで，ディレクトリー競合を回避し，パフォーマンス低下を回避することができると考えられる. 第 3 章のフローにあるように, SONAS 非同期コピーは，更新ファイルのリストを各 Int. ノードに分割して渡す. そこで，今回ファイル・リストの分割の仕方を見直すことにした. まず更新ファイル･リストの生成の仕方について述べ，その後，更新ファイル・リストの従来の分割方法，そして新しい分割方法について述べる.

6.1 更新ファイル・リストの生成および特徴

第 3 章のフローのステップ 2）において，更新ファイル・リストの作成を行う. 具体的には, GPFS の mmapplypolicy というコマンドの LIST 機能を使う. mmapplypolicy は，GPFS のメタデータをスキャンし，指

定時刻以降に更新のあったファイルをリストアップする．その際，リストは必ず i-node の順番になる．i-node は，GPFS ファイル・システムがファイルやディレクトリーを一意に特定するための識別子である．

更新ファイル・リストを分割する上で，その順番は重要なので，GPFS の i-node の割り振りについて調査を行った．調査結果を以下に示す．

- 各 Int. ノードが i-node の範囲を持っていて，ファイルやディレクトリーを作成する際に，その範囲から基本的に連続して番号を割り振っていく．
- 解放された番号の再利用が行われる．

従って通常多くのユーザーが多くの Int. ノードからファイル作成などを行った場合，i-node 番号はディレクトリーに関してはランダムである．ただし，データのマイグレーションやインストレーションなどで，ある 1 つのノードが，あるディレクトリー内のファイル群を大量に連続して作成した場合は，それらのファイルの i-node が連続している可能性がある．その場合は，i-node 順にリストされる更新ファイル・リストにおいて，同一ディレクトリーに含まれるファイル群が，連続してまとまっている可能性がある．

6.2 従来のファイル・リスト分割方法

従来，ファイル・リストを複数ノードに分割する際，リスト上のファイルのディレクトリーは考慮していなかった．各ノードが担当する数が均等になるように，ファイル・リストの先頭から 1 つずつ,ノードに配るようにして,分割していた（以下，シャッフル分割）．

6.3 新しいファイル・リスト分割方法

今回，ディレクトリーを考慮したファイル分割方法を取り入れた．

- GPFS スキャン機能により生成された更新ファイル・リストを，絶対パスでソートする．ただし，更新ファイル数が多い場合やファイル・パスが長い場合は，ソートに時間がかかってしまう．このソートにかかる時間と，新手法によりディレクトリー競合を回避した場合に短縮されるコピー時間の両方を考慮する必要がある．今回は，更新ファイル数に調整可能な閾値（以下，ソート閾値）を設け，その値以下の場合のみソートを行うようにした．
- 更新ファイル・リストを Int. ノード数で分割する．従来のように 1 つずつ先頭から配るように分けるのではなく，先頭から連続したまとまりで分割する（以下，ブロック分割）．

新しい分割方法の目的は，同じディレクトリーに含まれるファイル群をグループ化し，極力その固まりを 1 つの分割ファイル・リストのみにのせることである．結果として，その分割ファイル・リストを担当する 1 つの Int. ノードのみが，その同一ディレクトリーに含まれるファイル群の rsync を実行する.これにより,複数ノードによる同一ディレクトリーのトークンの競合を避けることができる．

更新ファイル数がソート閾値以下で,ファイル・リストのソートを行う場合は，確実に同一ディレクトリーのファイルをグループ化することができる．一方，更新ファイル数がソート閾値より大きい場合は，ソートを行わないので，確実にグループ化することはできない．しかし，今回，分割方法をシャッフル分割からブロック分割に変更したことにより，同一ディレクトリーに含まれるファイル群が，1 つの分割ファイル・リストのみにグループ化される可能性がある．なぜなら，6.1 章で述べたように，i-node 順にリストアップされる更新ファイル・リストでは，同一ディレクトリーに含まれるファイル群が，連続してリストされている可能性があるからである．

7. 新しいファイル・リスト分割方法の効果

新しい分割方法と従来の分割方法を用いた非同期コピーのパフォーマンス測定を行った．まずは，ディレクトリー数が少なく，ディレクトリー競合が起こりやすいテスト・ファイル構成における測定結果を図 6 に示す．

図から，新しい分割方法（3 ノード・新手法）が従来の方法（3 ノード・旧手法）に比べ最大 47% 短い時間でコピーを完了することが分かる．なおソート閾値は 200,000 ファイルに設定した．更新ファイル数がソート閾値以下で，ファイル・リストのソートを実行した場合は，ディレクトリー競合がなくなり，従来方法に比べ，パフォーマンスが最大 47% の時間短縮に向上している．1 Int. ノード（1 ノード）と比較しても，最大 37% の時間短縮の良いパフォーマンスが得られ,Int. ノード数の増加に伴い,パフォー

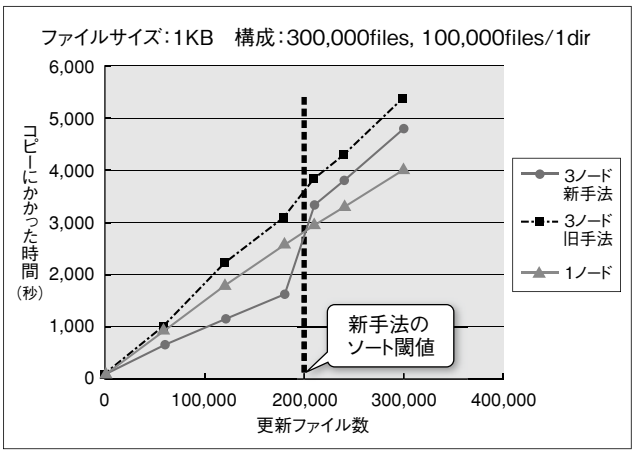


図 6. 新手法の効果

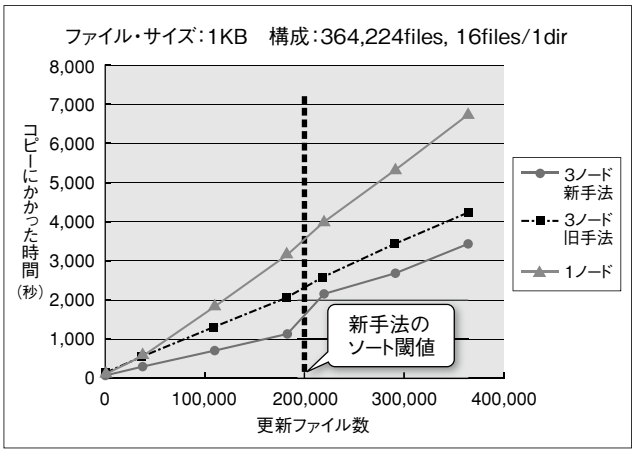


図 7. マルチ・ディレクトリー・テスト構成における新手法の効果

マンスが向上することが予想される.

さらに図 6 より，新しい分割方法の傾向は，ソート閾値を境に大幅に変化しているのが分かる．更新ファイル数がソート閾値よりも多く，ファイル・リストのソートを実行しなかった場合は，従来方法に比べ，ディレクトリー競合がやや減り，パフォーマンスが多少向上する程度（約 11% 時間短縮）にとどまった.

次に，ディレクトリー数が多く，比較的ディレクトリー競合が起こりにくいファイル構成（第 5 章マルチ・ディレクトリー・テスト構成）における新手法の効果を図 7 に示す.

新しい分割方法（3 ノード・新手法）と従来の方法（3 ノード・旧手法）を比較すると，このテストのようにもともと競合する確率が低い状況においても，新手法で確実にディレクトリー競合が起こらないようにすると，コピー時間をソート閾値以下において最大 48% 短縮することができた．また新しい分割方法の傾向は，やはりソート閾値を境に変化しており，ディレクトリー競合の影響の大きさを表し

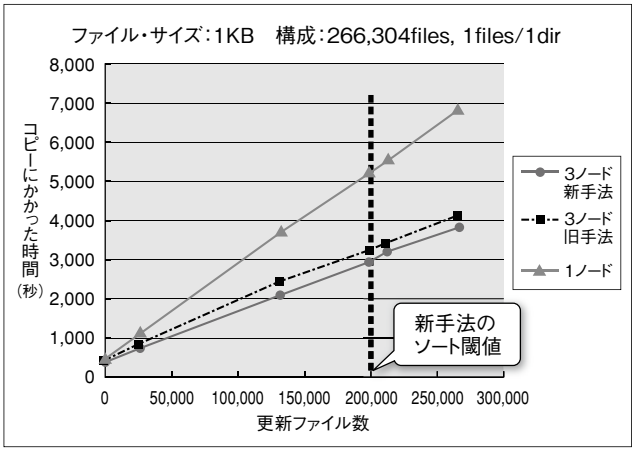


図 8. ディレクトリー競合が起こらない構成における新手法の効果

ている.

なお，6.3 章で述べたように，ソート閾値は調整可能になっている．今後実機 SONAS を用い，ソートにかかる時間とディレクトリー競合を回避することにより短縮される時間を比較して，最適なソート閾値を決めることが望ましい.

最後に，ディレクトリー競合以外のパフォーマンス影響因子の有無を確認するテストを行った．1 ファイル / 1 ディレクトリー構成のテストファイル群を作成し，理論上ディレクトリー競合が起こらないようにした．図 8 に結果を示す.

まず，ディレクトリー競合がないため，3 Int. ノード使用時の方が 1 Int. ノード使用時よりもパフォーマンスが良い．さらに，新しい分割方法と従来の分割方法でパフォーマンスにほぼ差がなかった．また新しい分割方法において，ソート閾値の前後で，パフォーマンスに変化はなかった．これらのことから，ディレクトリー競合によるペナルティーがパフォーマンスに影響を与える主な因子であり，それに対処する提案手法が効果的であることが証明された.

8. 今後の課題

4.2 章のベーシック・テストのような，更新ファイルのディレクトリーの数が Int. ノードの数よりも少ない場合，新しいファイル・リスト分割方法でも，ディレクトリー競合が必ず起きてしまう．この場合，rsync を実行する Int. ノードの数を制限するなどの実装が必要になる.

新しいファイル・リスト分割方法は，ファイル・リストのソートを行う．このソートは処理が重いため，今回は更新ファイル数に閾値を設けている．ソートの目的は，同一ディレクトリーのファイルをファイル・リスト内でグループ化することであり，グループ内のファイルの並び替えやグループの順番の並び替えは必要ではない．現在，ソートに代わる，より処理の軽い方法を検討中である.

9. おわりに

2010 年に IBM が発売した SONAS の非同期コピーでは，小さいファイルが数多く同一ディレクトリーに含まれるような構成の場合，複数ノードによる並列処理がパフォーマンスを大幅に劣化する問題が起きた．実験から GPFS のアクセスがボトルネックになっていることを確かめ，ディレクトリー・トークンの取得が競合していることが原因であることを明らかにした.

本論文では，そのディレクトリー競合を回避するために，SONAS の非同期コピー・モジュールに，更新ファイルのディレクトリーを考慮しつつ更新ファイル・リストを各 Int. ノー

ドに分割する新手法を提案した．本手法により従来方法に比べ，コピー完了にかかる時間を最大で 48% 短縮することを確認した．

この方法により，ある構成時の SONAS 非同期コピーのパフォーマンス問題が解消され，ノードの並列処理化によるパフォーマンスの向上が保証された．また，一回の非同期コピーにかかる時間を短縮することで，より短い RPO を実現でき，災害復旧ソリューションの信頼性を高めることが可能となった．本手法は 2011 年 5 月に発売した SONAS R1.2 に搭載されている．

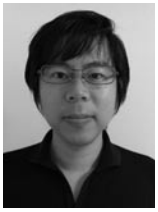
SONAS のように拡張性を最大の強みにする製品において，お客様の環境に適合したベスト・プラクティスの提案は重要であり，同時に難しい点でもある．今回の取り組みは，お客様の実環境とファイル・システムのもつ関連性・要因に着眼し，必要に応じて調整可能となる要素を組み込んだ．今後もパフォーマンスに影響を与えるファイル・システム要素の抽出および最適化を行いながら，ベスト・プラクティス・ガイドに反映したいと考えている．

謝辞

本論文の作成にあたり，IBM US および IBM Mainz の SONAS 非同期コピー開発チームの方々，ならびに IBM US の GPFS 開発チームの皆様に多くの助言をいただきました．あらためて深謝いたします．

参考文献

［1］IBM System Storage DS8000, http://www.ibm.com/systems/jp/storage/products/disk/ds8000/features.shtml
［2］IBM System Storage N7000 シリーズ, http://www.ibm.com/systems/jp/storage/products/nas/n7000/appliance/features.shtml
［3］日立ストレージソリューション ディザスタリカバリ, http://www.hitachi.co.jp/products/it/storage-solutions/solution/04/index.html
［4］IBM Scale Out Network Attached Storage（SONAS）http://www.ibm.com/systems/jp/storage/products/nas/sonas/
［5］IBM Redbooks "IBM System Storage DS8000: Copy Service in Open Environments", Fourth Edition（May 2008）.



日本アイ・ビー・エム株式会社
大和システム開発研究所
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．入社以来，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
大和システム開発研究所
JSSC プロジェクト・オフィス
XIV 開発　アドバイザリー
ソフトウェア エンジニア

萩原 克彦 Katsuhiko Hagiwara

［プロフィール］
1989 年，日本 IBM 入社．ホストエミュレータなどのソフトウェア製品の開発を経て，2001 年からテープドライブ製品のマイクロコード開発に従事．2009 年から，SONAS のコピーサービス開発のリードを行い，現在は XIV インターオペラビリティ開発に従事．

日本アイ・ビー・エム株式会社
大和システム開発研究所
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．同社大和システム開発研究所にて，SONAS のコピーサービスの設計・開発に従事．

日本アイ・ビー・エム株式会社
大和システム開発研究所
ストレージシステムズ開発
シニア　ソフトウェア開発マネージャ

岩崎 礼江 Norie Iwasaki

［プロフィール］
1990 年より HDD（ディスク装置），実時間 Mach マイクロカーネル研究開発，NAS，仮想テープ・サーバなどのストレージ製品制御ソフトウェアの設計・開発に従事．現在は VTS, SONAS 製品開発マネージャ．