マルチラック環境における HDFS の効率的なレプリカ再配置手法の提案と評価

日開　朝美† 　竹房あつ子†† 　中田　秀基†† 　小口　正人†

† お茶の水女子大学　〒 112-8610　東京都文京区大塚 2-1-1
†† 産業技術総合研究所　〒 305-8568　茨城県つくば市梅園 1-1-1
E-mail: †asami@ogl.is.ocha.ac.jp, ††{atsuko.takefusa,hide-nakada}@aist.go.jp, †††oguchi@computer.org

あらまし　Hadoop Distributed File System(HDFS) では，ノードが故障するとデータノード間でレプリカ再配置処理を行いデータの可用性を維持する．しかしながら，HDFS のレプリカ再配置では，レプリカ生成元・生成先をランダムに選択するため，データ移動に偏りが生じ非効率な再配置処理が行われている．この問題を解消するために，我々は指向性リング構造に基づいたデータ転送により各ノードの負荷を均衡化するレプリカ生成元・生成先の選出と，可用性とネットワーク負荷を考慮してスケジューリングを行う手法を提案し，2 ラックからなる HDFS クラスタにおいて，その特性及び性能が向上することをシミュレーションを用いて示してきた．2 ラック時には，ラック間転送の生成先のラックが一意に決定するのに対して，3 ラック以上の場合には生成先のラックを任意に選択することができ新たな制御が考えられる．本稿では既提案手法を拡張して，3 つ以上のラックからなる HDFS クラスタにおける効率的なレプリカ再配置手法を提案し，シミュレーションで評価する．評価実験より，提案手法により再配置の性能が向上し，最大で再配置の実行時間が 20%削減できることを示す．

キーワード　HDFS, 分散ファイルシステム, レプリカ, 再配置

An Evaluation of Effective Replica Reconstruction Schemes for Multi-rack HDFS Environments

Asami HIGAI†, Atsuko TAKEFUSA††, Hidemoto NAKADA††, and Masato OGUCHI†

† Ochanomizu University　2-1-1 Otsuka, Bunkyouku Tokyo, 112-8610 JAPAN
†† National Institute of Advanced Industrial Science and Technology (AIST)
1-1-1 Umezono, Tsukuba, Ibaraki, 305-8568 JAPAN
E-mail: †asami@ogl.is.ocha.ac.jp, ††{atsuko.takefusa,hide-nakada}@aist.go.jp, †††oguchi@computer.org

1. はじめに

近年，センサネットワークやソーシャルメディアの普及により，データが日々刻々と生成されるようになり，科学技術分野や商業分野への活用を始めとし，大規模データを効率良く管理，処理することが求められている．このような大規模データに対応した処理システムとして，汎用的なハードウェアを用いて高度な集約処理を可能にする分散ファイルシステムが広く利用されている．分散ファイルシステムは，データに対して複数のレプリカを生成し，多数のノードを用いて分散管理することで可用性や耐故障性を維持している．ノードが故障すると，そのノードが管理していたレプリカが一時的に不足し，そのデータを保持している他のノードへのアクセス負荷が増加して，システム全体の性能が低下する．そのため不足レプリカの再配置を高速に行い，データ処理システム全体の性能低下を防ぐことが重要である．

オープンソースの分散ファイルシステムでは，Apache Hadoop(Hadoop) [1] プロジェクトの Hadoop Distributed File System(HDFS) [2] が広く用いられている．しかしながら HDFS のレプリカ再配置では，生成元・生成先をランダムに選択するため，データ移動に偏りが生じ，非効率なレプリカ再配置処理が行われている．我々はこの問題を解消するために，指向性リング構造に基づいたデータ転送により各ノードの負荷を均衡化する生成元・生成先選出アルゴリズムと，可用性とネットワーク負荷を考慮したスケジューリング手法を提案し，2 ラックからなる HDFS クラスタにおいて，その特性及び性能が向上す

ることをシミュレーションを用いて示した [3]．しかしながら，3つ以上のラックから構成されるより大規模な HDFS 環境での再配置処理は十分に検討されていない．

本稿では 3 つ以上のラックからなる HDFS クラスタにおける効率的なレプリカ再配置手法を提案し，シミュレーションで評価する．2 ラック時には，ラック間転送時の生成先ラックが一意に決定していたのに対して，3 ラック以上の場合には複数のラックから選択可能になるため，新たな転送制御が可能となる．指向性リング構造に基づいてデータ転送を行う既提案手法を応用して，ラック内とラック間それぞれにリング構造を作成し，データ転送を行うものとし，また各ラックの送受信ブロック数を均衡化するように処理の割り当てを定式化し，それらに基づいて生成元・生成先を選出する制御手法を提案する．SimGrid シミュレータ [4] を用いた評価実験から，提案手法によりデータ移動の偏りが解消され，再配置の実行時間を最大で 20%削減することができた．

2. HDFS のレプリカ再配置

HDFS は Apache Hadoop プロジェクトの一つで Google の Google File System(GFS) [5] に基づいて設計された分散ファイルシステムである．HDFS はマスタ・ワーカ型の構成であり，ファイルのメタデータやクラスタ内のノード管理を行う一台の NameNode と，実際にデータを格納し処理を行う複数台の DataNode からなる．ファイルを最小単位であるブロックに分割し，各ブロックに対して複数のレプリカを複数の DataNode 間で分散して保存することで高い読み書き性能とデータの可用性，システムの耐障害性を提供している．

2.1 レプリカ配置ポリシー

HDFS は一般にデータを管理するために多数の DataNode を用いて，マルチラック構成のクラスタで運用される．この場合，レプリカは信頼性と負荷分散を考慮して 2 つ以上のラックにレプリカが配置され，2015 年 2 月時点の最新版の v.2.6.0 では，以下のようにレプリカを配置する．

第 1 レプリカ　書き込みがクラスタ内部で開始される場合にはローカルの DataNode，そうでない場合にはクラスタ内からランダムに選ばれた DataNode

第 2 レプリカ　第 1 レプリカとは異なるラックの DataNode

第 3 レプリカ　第 2 レプリカと同じラック内の異なる DataNode

それ以降のレプリカ　任意のラックの任意の DataNode

2.2 レプリカ再配置処理

DataNode が故障した場合にはその DataNode が保持するレプリカがなくなるため，残りの DataNode 間でその不足レプリカを補うレプリカ再配置処理が行われる．レプリカ配置ポリシーに基づき，生成元・生成先をランダムに選択し，適当なラックにレプリカを再配置する．デフォルトの設定であるレプリカ数 3 を例にすると，DataNode が故障した際のレプリカの配置は，異なるラックに残りのレプリカが存在する場合と同一ラックに残りのレプリカが存在する場合のどちらかである (図 1)．HDFS クラスタでは，経由するスイッチが増加する分ネットワーク帯域幅が減少するという前提の下，ラック間の転送よりラック内の転送を優先する．そのため異なるラックに残りのレプリカが存在する場合には，同一ラック内にレプリカを再配置するラック内転送が発生する．一方，同一ラックに残りのレプリカが存在する場合には，異なるラックにレプリカを再配置しなければならないので，ラック間の転送が発生する．この時各 DataNode が同時に送信できるストリーム数は 2 本に制限される．

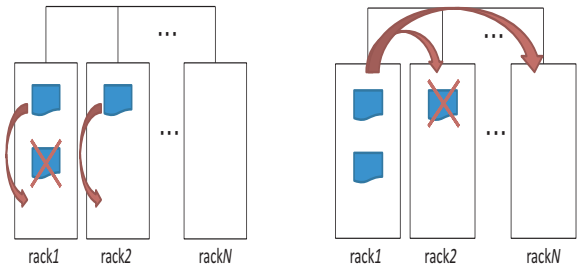

図 1　左：異なるラックに残りのレプリカが存在する場合 (ラック内転送)
右：同一ラックに残りのレプリカが存在する場合 (ラック間転送)

2.3 マルチラック環境におけるレプリカ再配置の課題

レプリカ生成元・生成先をほぼランダムに選出するデフォルトのレプリカ再配置手法を用いて，ある 1 つのラックのうちの 1 台の DataNode を削除した際のレプリカ再配置処理を，4. 章に述べるシミュレーション環境にて，ラック数を 3，1 ラックあたりの DataNode 数を 8 台として調査した．この時の各ラックの送受信のブロック数と，各転送形態の 1 ブロックあたりの平均転送時間を表 1，2 に示す．ここで，削除ノードを含むラックを failure rack，削除ノードを含まないラックを normal rack と呼ぶ．表 1 より，failure rack C はラック間転送の生成元ノードにはなり得ないが，生成先ノードには選出されるため，受信ブロック数が圧倒的に多くなっていることが分かる．また表 2 より，failure rack C に関連する転送の転送時間が長くなっていることが分かる．以上よりマルチラック環境においては，failure rack のノードに転送が集中する傾向にあり，その結果 failure rack に関する転送時間が長引く事態が発生し，非効率なレプリカ再配置が行われていることが分かった．

3. マルチラック環境におけるレプリカ再配置手法の提案

3 ラック以上の HDFS 環境においても効率良くレプリカ再配置を行うために，2 ラック環境におけるレプリカ再配置の既提案手法 [3] の指向性リング構造に基づくデータ転送及び各 DataNode の送受信のブロック数を均衡化させる制御を応用して，レプリカ生成元・生成先の選出制御と，スケジューリング制御の提案を行う．

3.1 レプリカ生成元・生成先の選出

各 DataNode の送信と受信のブロック数を等しくするために，failure rack はラック間転送に関与させず，ラック内転送のみを行うものとし，その分ラック内転送の負荷を増加させることにより，効率的なレプリカ再配置を実現するレプリカ生成

表 1　各ラックの送受信のブロック数 (デフォルト手法)

		normal rack		failure rack
		rack A	rack B	rack C
ラック間転送	送信	281	334	0
	受信	180	154	281
ラック内転送	送信	402	370	453
	受信	402	370	453
送信合計		683	704	453
受信合計		582	524	734
送受信合計		1265	1228	1187

表 2　1 ブロックあたりの平均転送時間

		転送時間 [sec]
ラック間転送	rack A to rack B	4.138
	rack A to rack C	5.817
	rack B to rack A	4.419
	rack B to rack C	5.378
ラック内転送	rack A	4.549
	rack B	4.247
	rack C	6.211

表 3　各ラックへの再配置処理の割り当て

	normal rack	failure rack
ラック間転送	$B_{inter} \times \frac{1}{R-1}$	0
ラック内転送	$\left(B_{inner} - \frac{B}{R}\right) \times \frac{1}{R-1}$ $= B_{inner} \times \frac{2R-3}{2R(R-1)}$	$B \times \frac{1}{R}$ $= B_{inner} \times \frac{3}{2R}$

元・生成先の選出制御手法を提案する．

ラック数を $R(\geq 3)$ とし，1 ラックあたりの DataNode 数が等しく，データが一様に分散配置されているとすると，1 台の DataNode が削除された際，複製の必要なブロック数を B とすると，確率的にラック間転送が行われるブロック数 $B_{inter} = \frac{1}{3}B$，ラック内転送が行われるブロック数 $B_{inner} = \frac{2}{3}B$ である．各ラックが $\frac{B}{R}$ 個のブロックを送受信すると処理ブロック量が等しくなるため，それぞれのラックに表 3 のように再配置処理を割り当てる．

生成元 DataNode の選出における，ラック間転送では，生成元候補ノードの中からラック間生成元選出回数が最小の DataNode を選出することで，表 3 を満たすことが出来る．ここで生成元候補ノードとは，該当の不足レプリカの残りのレプリカを保持する DataNode を指す．ラック内転送では，表 3 の処理の割り当てを考慮し，生成元候補ノードのラック内生成元選出回数を，normal rack の場合は $3(R-1)$ 倍，failure rack の場合は $2R-3$ 倍して，比較を行い，選出回数が最少となる DataNode を選出する．生成先 DataNode の選出は，既提案手法の指向性リング構造に基づいて 1 つ先の DataNode を選出する手法を応用する．図 2 に示すように，ラック内転送のためにラック毎に論理的なリング構造を構成し，ラック間転送のために normal rack に含まれる全ての DataNode を繋ぐ 1 つの論理的なリング構造を作成する．この時，ラック間転送のリング構造に関して前後の DataNode は異なるラックに属する DataNode となるようにする．このようなリング構造により，ラック間及びラック内転送それぞれにおいて，生成元が決定すると一意の DataNode が生成先として決定する．そして表 3 に基づく生成元の選出により，各 DataNode の送信ブロック数が均衡化され，それに伴い受信ブロック数も付随して均衡化される．

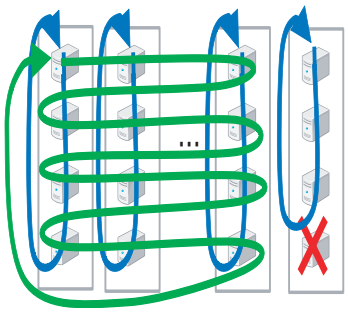

図 2　論理的な指向性リング構造

3.2　スケジューリング制御

前節のレプリカ生成元・生成先選出制御手法をベースとして，レプリカ転送順を制御する「優先度付手法」と制御を行わない「優先度無手法」を提案する．スイッチや電源系統の故障など，万が一レプリカ再配置中にラック全体に渡る障害が発生した場合，同一ラックに残りのレプリカが存在するブロックは復元不可能になってしまう．そのため，ラック間転送が必要なブロックを先に再配置することは可用性の向上に繋がる．そこで，上記のレプリカ生成元・生成先ノードの選出により，生成元・生成先ノードが決定した不足レプリカに対して，同一ラックに残りのレプリカが存在する，即ちラック間転送を行うブロックに高い優先度を付与して先にスケジューリングした後に，異なるラックに残りのレプリカが存在する，即ちラック内転送を行うブロックをスケジューリングする制御を行う．これを優先度付手法と呼ぶ．優先度付手法は可用性の向上を期待した手法である．一方，これら 2 つの状態のブロックを区別することなく，任意の順にスケジューリングする制御手法を優先度無手法と呼ぶ．

4.　評 価 実 験

HDFS のデフォルト再配置手法と提案する優先度無手法，優先度付手法を用いて，ある 1 つのラックのうちの 1 台の DataNode を削除した際のレプリカ再配置処理の性能をシミュレーションで評価する．評価では以下を調査する．

(1)　レプリカ再配置の実行時間
(2)　1 ブロックあたりの平均転送時間
(3)　ラック間転送を伴うブロックの複製の進行度

(2) はブロックの複製が効率良く行われているか否かを各再配置手法毎に評価する．(3) では各手法の可用性を評価する．

実験には，離散イベントシミュレータの SimGrid-3.10 [4] を用い，表 4 に示すパラメータを用いた．ディスク性能は，1 対 1 のレプリカ再配置時のディスク I/O の実測値が 67MB/sec であることからこの値に設定した．

表 4　シミュレーション実験に用いたパラメータ

ラック数	3，4，5
1 ラックあたりの DataNode 数	8, 16, 32，64
ブロックサイズ	67MB(default)
レプリカ数	3(default)
不足ブロック数 (削除ノードが保持するブロック数)	80*正常な DataNode 数
ラック内ネットワーク帯域幅	125 MB/sec
ラック内ネットワーク遅延	0.1 msec
ラック間ネットワーク帯域幅	1.25 GB/sec
ラック間ネットワーク遅延	0.1 msec
ディスク性能	67 MB/sec

4.1　評価結果

各手法のレプリカ再配置の実行時間を図 3，4，5 に示す．図 3，4，5 より，ラック数の違いによる大きな変化はなく，1 ラックあたりの DataNode 数が 64 台の場合を除き，提案手法によりレプリカ再配置の実行時間が減少し，優先度無手法では最大で 12%，優先度付手法では最大で 20%実行時間を削減できた．これは，failure rack への処理集中の回避，送受信のブロック数の均衡化，ラック内及びラック間の指向性リング構造に基づくデータ転送により，効率良く処理が行われたためだと考えられる．優先度無手法より優先度付手法の方が性能が高い理由は，優先度無手法は，任意の順にスケジューリングを行うためラック間転送とラック内転送が混在し，指向性リング構造に基づいたデータ転送であっても，ある DataNode に関して最大でラック内転送・ラック間転送それぞれ 2 つずつの計 4 つのブロックを受信してしまう事態が発生し転送に時間がかかるのに対し，優先度付手法は，normal rack のデータ転送に関して，処理の冒頭はラック間転送のみが行われ，それらが終了した後にラック内転送が実行されるため，時系列的にみると生成元と生成先が一対一に対応し最大受信ブロック数が 2 に制限された転送が行われ，より効率良く処理が行われるからである．1 ラックあたりの DataNode 数が 64 台の場合には，実行時間が大幅に増加している．これは DataNode 数が多いために，ラック間転送を行うブロックに高い優先度を付与する優先度付手法では，ラック間のネットワーク帯域を超えたデータ転送が処理の序盤で行われてしまうためである．特に本実験環境においては，ディスク性能が 67MB/sec，ラック内のネットワークが 125MB/sec，ラック間のネットワークが 1.25GB/sec であり，読み書きがディスク性能を 2 分し，ラック内の全ての DataNode がラック間転送を行うとすると，67MB/sec÷2×(DataNode 数) のデータ量がラック間のネットワークに流れ，ラック内の DataNode 数が 38 台以上になるとラック間のネットワークがボトルネックになる．デフォルト手法と優先度無手法では，ラック間転送及びラック内転送のブロックを任意の順にスケジューリングするため，1 ラックあたりの DataNode 数が 64 台の場合でも，ラック間ネットワークの帯域を超えたデータ量が流れることはあまりないため顕著な性能低下は見られないが，優先度付手法では，処理の序盤には normal rack の全ての DataNode がラック間転送を行うため，ラック間ネットワークの帯域を超えたデータが

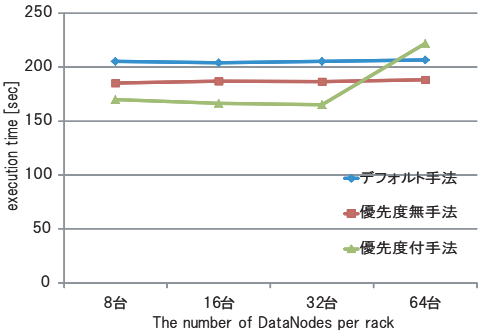

図 3　各手法におけるレプリカ再配置の実行時間 (ラック数 3)

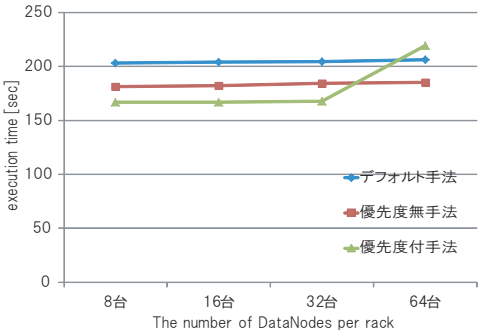

図 4　各手法におけるレプリカ再配置の実行時間 (ラック数 4)

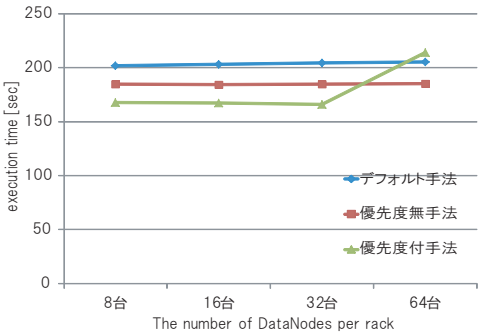

図 5　各手法におけるレプリカ再配置の実行時間 (ラック数 5)

流れ性能が低下している．

1 ブロックあたりの平均転送時間を，ラック数が 3，1 ラックあたりの DataNode 数が 8，64 台の場合をそれぞれ表 5，6 に示す．ラック数が 4，5 の場合も同様の傾向のため，ここでは省略する．表 5 より，デフォルト手法では failure rack C に関連する転送時間が長くなっていたが，提案手法，特に優先度付手法では，全ての転送形態についてほぼ等しい転送時間が実現されており，効率良く処理が行われていることが分かる．表 6 より，1 ラックあたりの DataNode 数が 64 台と，ラック間通信が飽和する場合には，優先度付手法のラック間転送の転送時間が長くなっていることが確認できる．

各手法におけるラック間転送を伴うブロックの複製の進行度合い [%] を，ラック数が 3，1 ラックあたりの DataNode 数が 8，64 台の場合をそれぞれ図 6，7 に示す．1 ラックあたりの DataNode 数が 16，32 台の場合は 8 台の場合と，ラック数が 4，5 の場合はラック数が 3 の場合と同様の傾向のため，ここ

表 5　1 ブロックあたりの平均転送時間 [sec]
(ラック数：3，1 ラックあたりの DataNode 数：8)

		デフォルト	優先度無	優先度付
ラック間転送	rack A to rack B	4.138	4.635	4.032
	rack A to rack C	5.817	0	0
	rack B to rack A	4.419	4.509	4.267
	rack B to rack C	5.378	0	0
ラック内転送	rack A	4.549	4.544	4.282
	rack B	4.247	4.210	3.903
	rack C	6.211	3.988	3.988

normal rack：rack A，rack B　　failure rack：rack C

表 6　1 ブロックあたりの平均転送時間 [sec]
(ラック数：3，1 ラックあたりの DataNode 数：64)

		デフォルト	優先度無	優先度付
ラック間転送	rack A to rack B	4.242	4.591	6.795
	rack A to rack C	5.799	0	0
	rack B to rack A	4.236	4.590	6.814
	rack B to rack C	5.851	0	0
ラック内転送	rack A	4.222	4.397	3.940
	rack B	4.178	4.409	3.981
	rack C	6.295	3.991	3.991

normal rack：rack A，rack B　　failure rack：rack C

では省略する．図 6 より，1 ラックあたりの DataNode 数が 8 台の場合には，優先度付手法のラック間転送を伴うブロックの複製が非常に早く完了し，可用性が向上していると言える．一方で図 7 より，1 ラックあたりの DataNode 数が 64 台の場合には，8 台のときと比較すると優先度付手法のラック間転送を伴うブロックの複製が長期化している．これは上述したように，DataNode 数が多くなるとラック間ネットワークが飽和してしまうことが原因である．

以上より，1 ラックあたりの DataNode 数が飽和しない場合には，優先度付手法は高い可用性を提供しながら，高速にレプリカ再配置を実行できる有効な手法であることが分かる．また，ラック間通信が飽和する場合は，3 ラック以上の場合においてもストリーム制御を適切に行うことが重要であることが示された．

5. 関 連 研 究

分散システムにおいて，データのレプリカをどのように管理するかというレプリケーション手法が数多く提案されている．一般的にデータを複製して、複数のレプリカを複数の異なるマシン上に保存することで、システム性能や耐故障性、信頼性を提供しているが，この時，レプリカ数とレプリカの配置場所が重要になってくる．

ファイルクラスタリングに基づいて，アクセス時間やストレージ容量などの性能要件を満たし，レプリカの複製時間が最小になるようにレプリカの配置を決定する手法 [6] [7] [8] や，ファイルの人気度に基づいて，レプリカ配置場所と複製数を決定する手法 [9]，アクセス負荷に応じて動的にレプリカの複製を行う手法 [10]，単一のリクエストのレスポンスタイムに注目

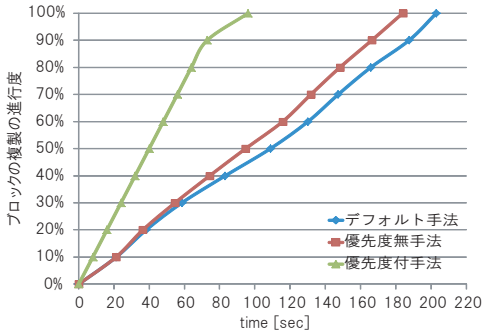

図 6　各手法におけるラック間転送を伴うブロックの複製の進行度
(ラック数：3，1 ラックあたりの DataNode 数：8)

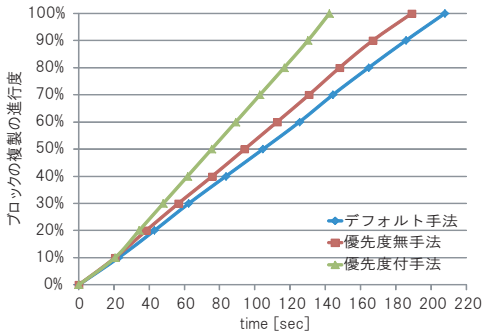

図 7　各手法におけるラック間転送を伴うブロックの複製の進行度
(ラック数：3，1 ラックあたりの DataNode 数：64)

した手法 [11] など，数多くのレプリカ配置手法が提案されている．多くの提案手法は，アクセス時間やストレージ容量，複製時間を考慮したレプリカ配置手法であり，我々が提案したような，複製処理自体が各ノードに与える負荷についてはあまり考慮されていない．

一方で鈴木ら [12] は，広域環境におけるクラスタ間で複数のファイルの複製を効率的に行う手法について考察し，アルゴリズムの提案と評価を行っている．クラスタ間で効率良くファイルを複製するには，単一ディスクへのアクセス集中による性能低下とネットワークの通信性能低下の回避が重要であると述べ，適切なノードにファイル複製を割り当てる「ファイル複製選択アルゴリズム」とそれらを適切なコネクションに割り当てる「転送順序スケジューリングアルゴリズム」を提案している．前者のアルゴリズムを線形計画法及び貪欲法を用いて解き，後者をリストスケジューリング法を用いて解く手法をシミュレーションにより評価し，それらの提案手法が有効であることを示している．適切な複製元選択とスケジューリングは本研究に通ずるが，複製先のノードも選択しなければならない点と，複製先が同一ラック内の場合もある点が本研究と異なる．

6. ま と め

3 つ以上のラックからなる HDFS のレプリカ再配置において、各ラックの送受信ブロック数に偏りが生じ，特に削除ノードを含むラックへの処理集中により，非効率なデータ転送が行われるという問題を解決するために，レプリカ生成元・生成先

の選出制御手法を提案した．提案するレプリカ生成元・生成先の選出では，削除ノードを含むラックはラック間転送を行わないようにした上で，各ラックの送信ブロック数を均衡化するように定式化してその比率に基づき生成元ノードを選出し，ラック内転送及びラック間転送それぞれの指向性リング構造に基づき，生成先ノードを一意に決定する．さらに可用性の向上を目的として，レプリカ生成元・生成先が決定したブロックに対して優先度をつけるスケジューリング制御を加えた手法を提案し，シミュレーションで評価した．評価実験から，提案手法により，データ移動の偏りが解消され，最大で再配置の実行時間を 20%削減することができた．本実験環境では，優先度付手法はラック内の DataNode 数が 64 台と多い場合にはラック間ネットワーク帯域が飽和して性能が低下してしまうが，32 台以下の場合には可用性の向上だけでなく，再配置処理を高速に行える，より有効な手法であることが分かった．

文　　献

[1] Tom White, Hadoop: The definitive guide, trans. Ryuji Tamagawa. O'Reilly JAPAN, 2010.

[2] Dhruba Borthakur. "HDFS Architecture," 2008 The Apache Software Foundation.

[3] Asami Higai, Atsuko Takefusa, Hidemoto Nakada, Masato Oguchi, "A Study of Effective Replica Reconstruction Schemes for the Hadoop Distributed File System," IEICE Trans. Inf. & Syst., Vol.E98-D,No.4,Apr. 2015 (To be appeared)

[4] SimGrid. http://simgrid.gforge.inria.fr/

[5] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung (October 2003), "The Google File System," 19th Symposium on Operating Systems Principles (conference), Lake George, NY: The Association for Computing Machinery, CiteSeerX: 10.1.1.125.789, retrieved 2012-07-12.

[6] Rashedur M.Rahman, Ken Barker, Reda Alhajj, "Study of Different Replica Placement and Maintenance Strategies in Data Grid," In Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid, pp.171-178, 2007.

[7] Y. Wang and D. Kaeli, "Load balancing using grid-based peer-to-peer parallel I/O," In Proceedings of IEEE International Conference on Cluster Computing, pp.1-10, 2005.

[8] Hitoshi Sato, Satoshi Matsuoka, and Toshio Endo, "File Clustering Based Replication Algorithm in a Grid Environment," In Proceedings of the 9th IEEE International Symposium on Computing and the Grid (CCGrid2009), pp.204-211, Shanghai, China, May 2009.

[9] K. Sashi, Antony Selvadoss Thanamani, "A New Replica Creation and Placement Algorithm for Data Grid Environment," International Conference on Data Storage and Data Engineering, pp.265-269, 2010.

[10] J. Tjioe, R. Widjaja, A. Lee, and T.Xie, "DORA:A Dynamic File Assignment Strategy with Replication," International Conference on Parallel Processing 2009.

[11] W.F. Wang, W.H. Wei, "A Dynamic Replica Placement Mechanism Based-on Response Time Measure," Proc. of IEEE International Conf. on Communications and Mobile Computing, pp.169-173, 2010.

[12] 鈴木克典，建部修見，"PC クラスタ間ファイル複製スケジューリング "情報処理学会論文誌コンピューティングシステム Vol.3 No.3 pp.113-125