クラスタ間並列複製作成のためのファイル分割を許さないスケジューリング

鈴木克典[†] 建部修見[†]

ネットワークの広帯域化に伴い遠隔 PC クラスタ間での大規模データ共有の要求が高まっている．一方，大規模データ処理のためにローカルディスクが積極的に用いられている．その結果，各 PC クラスタの多数のローカルディスク間における効率的な並列ファイル複製複製技術が必要となってきた．そこで我々は過去 PC クラスタ間の転送ファイル複製の選択，転送順序スケジューリング，実行時複製作成スケジューリングのためのアルゴリズムの提案を行った．しかし，スケジューリングアルゴリズムの対象問題は NP 困難であり，その緩和のためにファイル分割を許していたため，実環境での使用には問題があった．本論文ではファイル分割を許さない転送ファイル複製選択，実行時複製作成アルゴリズムを提案し，シミュレーションによって評価する．その結果，提案するファイル分割を許さないアルゴリズムは，対象問題が NP 困難であるにももかかわらず，設定した評価ではファイル分割を許す先行アルゴリズムに比べて転送時間の差は約 5%以内，計算時間の差は 2%以内に収まることが分かった．

A Replication Scheduling Between PC Clusters Without File Segmentation

Katsunori Suzuki[†] and Osamu Tatebe[†]

Thanks to increases in wide-area network bandwidth, it is now possible to efficiently share data on a large scale. While network bandwidth has become wider, network data access is still significantly slower than using local disks because of enormous latency. Therefore in order to improve I/O performance each compute node should read data from its local disk as much as possible. In order to achieve this goal, we will need efficient and parallel file transfer mechanisms between computation nodes. Thus, to improve the parallel file transfer performance between PC clusters, we proposed three algorithms; a file replica selection algorithm, a transfer ordering scheduling algorithm and a file replica creation algorithm. Although, our algorithm might pose a matter that create segmented file. The cause of the matter was permitting to segment files. But, the target problem of these algorithms is NP-hard. Therefore, we had to permit file segmentation to relax and solve the problem. In order to avoid the file segmentation problem, we develop new scheduling algorithms without file segmentation. Our simulation results show the ratio of file transfer time produced by our scheduling with and without file segmentation is less than 5%, and also the ratio of each scheduling of calculation time is less than 2%.

1. はじめに

科学技術分野では実験やシミュレーションにより多量のデータを生成し，その分析を必要とする．そこでネットワークが広帯域化するに伴い，世界各地の研究機関において大規模データを共有し研究の効率化が進められている[1),3)]．このような理由から広域環境下で大規模なデータを共有する要求が高まっている．実際に欧州原子核機構 (CERN) では膨大なデータの処理のためにヨーロッパ規模，世界規模のデータ・コンピューティンググリッドの利用が始まっている[8)]．

一方，近年では大規模データ処理計算において，PC クラスタによる並列計算が一般的に用いられている．PC クラスタで大規模データ処理を行う場合，ディスク I/O がボトルネックとならないように，NFS のような共有ファイルシステムではなくローカルディスクを効率的に利用することが重要になる．しかし，以上で述べたようにローカルディスクを活用した場合では，データが複数のストレージに分散されるてしまう．これらのデータの複製を他の PC クラスタに作成することを考えた場合，複数ディスクからの並列転送を行うことで効率的に複製作成を行うことができる．

そこで我々は文献[18)] において，高遅延・広帯域ネットワークで接続された PC クラスタ間で各計算ノードが並列にデータ転送を行うことを想定し，そのための

† 筑波大学大学院システム情報工学研究科
University of Tsukuba, Graduate School of Systems and Information Engineering

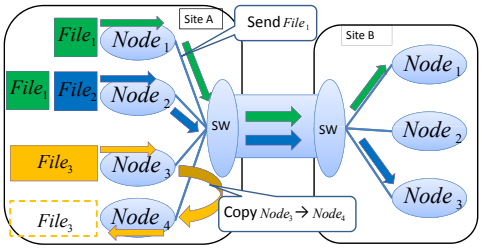


図 1 クラスタ間広域ファイル複製作成

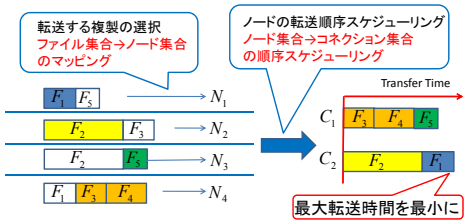


図 2 スケジューリングアルゴリズムの概要

ファイル転送スケジューリングを提案した．しかしながら，提案スケジューリングアルゴリズムは問題のクラスを簡単にするために，ファイルの分割を許し，複数の転送ノードに部分的に割り当てられることを認めていた．このとき，もとはひとつのファイルであったにもかかわらず作成された複製は複数のノードに分割された状態で格納されてしまう可能性があるなどの問題があった．そのため，本研究ではファイルの分割を許さない効率的なファイル転送スケジューリングアルゴリズムについて考察する．

本論文では，2 章において 18) で提案した PC クラスタ間広域ファイル複製作成についてその提案アルゴリズムの概要を述べ，問題点を考察する．次に 3 章で 2 章でまとめた問題点を解決するためのアルゴリズムについて述べた後，4 章では本論文で提案するスケジューリングアルゴリズムによって求められたファイル転送スケジュールの Rake ツールによる記述，転送処理実行について言及する．5 章では 18) と本研究の提案アルゴリズムのシミュレータによる比較評価や実環境における PC クラスタ間並列複製作成の評価について議論する．6 章で関連研究について言及し，7 章でまとめる．

2. クラスタ間広域ファイル複製作成

本節では 18) で提案した PC クラスタ間並列複製作成とスケジューリングアルゴリズムの概要について述べる．そして，それらを実際のファイル複製に適用した場合の問題点について考察する．

2.1 提案する並列複製作成の概要

提案手法は**図 1** で示すように複数の計算ノードを持ち，広帯域なネットワークで接続された 2 つの PC クラスタを想定する．また，2 つの PC クラスタ間のネットワークは高遅延であり，ノード間通信において PC クラスタ内でのデータ転送の方が PC クラスタ間でのデータ転送よりも高速であるとする．

このような環境において，例えば図 1 中の $Site\ A$ の各計算ノードのローカルディスクに複数のファイル，ここでは $File_1$，$File_2$，$File_3$ が格納されており，これらの複製を遠隔の PC クラスタ $Site\ B$ の各計算ノードのローカルディスクに作成することを考える．このとき，$File_1$ のようにファイルは送信元クラスタ内の複数のノード上に複製を持つ可能性がある．そのため，特定のノードからのファイル転送が全体のネックとならないように適切な複製を選択する必要がある．

しかし，このときひとつのノードから同時に複数のファイル転送を行った場合や，ひとつのノードへ同時に複数のファイルを転送した場合ではディスクのシークが頻発する．また，各ノードが同時にバースト転送を行うことで多量のネットワークトラフィックを発生させた場合，輻輳が発生する．そのため，転送スケジューリングでは，転送複製，同一ノード複数ファイルが同時にアクセスされないようなスケジューリング，および輻輳が発生しないような同時転送ファイルのスケジューリングを行う．[16]

また，提案複製作成手法では 2 ノード間において送信元クラスタ内と 2 クラスタ間の転送速度に差があることを利用し，送信元クラスタ内において積極的にファイル複製を作成したのちに並列転送を行うことでより効率的な複製作成を行う．

2.2 スケジューリングアルゴリズムの概要

2.1 節で述べたようにスケジューリングアルゴリズムは転送トラフィックの制御，各ノードが同時に複数のファイル送受信を行わないようなスケジューリング，適切な転送複製の選択を実現する必要がある．スケジューリングアルゴリズムの動作概要を**図 2** に示す．まず，転送トラフィックの制御を行うために特定バンド幅でトラフィックシェーピングを行った通信ストリームをコネクション C_i とし，コネクション C_i を通してファイル転送を行う．これら複数コネクションを束ねることでトラフィック制御を行ったうえでネットワークの効率的利用を行う．

次に，すべてのノード N_j に対し各ノード N_j が転送するファイル複製 F_k の選択割り当てを行う．その

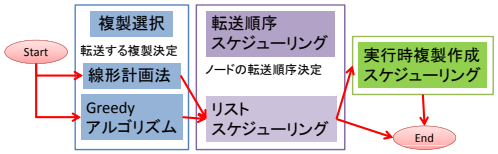


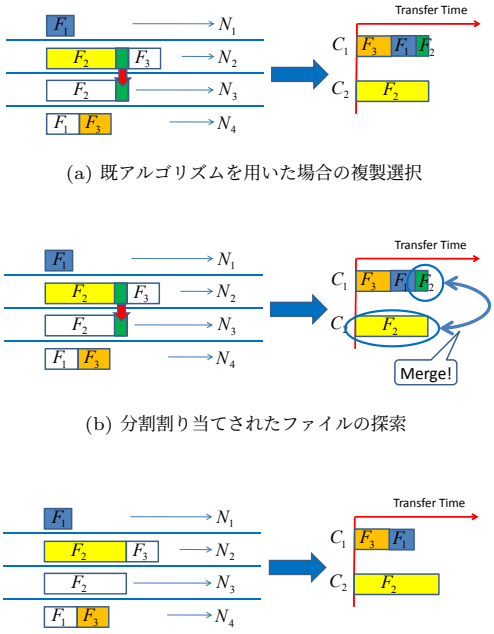
図 3 スケジューリングアルゴリズムの構成図

後，各ノード N_j ごとにコネクション集合 C に対し順序のスケジューリングを行う．最後に転送時において各ノード N_j は，割り当てられたファイル群を同時に送信しないことで，各ノード N_j が複数のファイルを同時に送受信しないことと，全体として適切な転送複製の選択を行ったスケジューリングを実現する．

以上で述べたように提案アルゴリズムはファイル集合 F からノード集合 N へのマッピングを行う複製選択フェーズ，ノード集合 N のコネクション集合 C への割り当て順序スケジューリングフェーズと 2.1 節で述べた実行時の送信元クラスタ内複製作成のスケジューリングフェーズの 3 フェーズ（**図 3**）からなる．ノード集合 N からコネクション集合 C への割り当て順序スケジューリングの問題は独立なタスクのスケジューリング問題[6),17)] に還元することができ，この問題は NP 困難である．そこで，このフェーズではリストスケジューリングアルゴリズム[9)] を用いて近似解を求める．また，リストスケジューリングではスケジューリングされるタスクがすべて均一であるならば良い近似解が期待できる．そのため，転送複製の選択フェーズではすべてのノード N_j に対し割り当てられるファイル F_i の総データ量ができる限り等しくなるように線形計画法や貪欲法を用いてファイル割り当てを求める．

2.3 スケジューリングアルゴリズムの問題点

1 節において先行アルゴリズムでは，もともとひとつであったファイルが，複製作成後は転送先クラスタで複数ノードに分割されたファイルとして格納される問題があることを述べた．これは，転送複製選択フェーズにおいて，求めるファイル集合 F とノード集合 N のマッピング問題は NP 困難な組み合わせ最適化問題となることから，問題の簡単化のために導入した「ファイルは分割してもよい」という仮定によっている．この仮定により，ファイル集合 F とノード集合 N のマッピング問題は各ファイルの各ノードへの割り当て配分を求める問題となり線形計画問題として多項式時間で解くことができる．また，この仮定は実行時複製作成フェーズにおいても利用されているため 1 節において述べた問題を解決するためには転送複製選択と実行時複製作成のアルゴリズムの改善が必要となる．



(a) 既アルゴリズムを用いた場合の複製選択

(b) 分割割り当てされたファイルの探索

(c) 分割割り当てされたファイルの修正・マージ

図 4 転送ファイル複製の選択におけるファイル分割割り当て回避

3. アルゴリズム

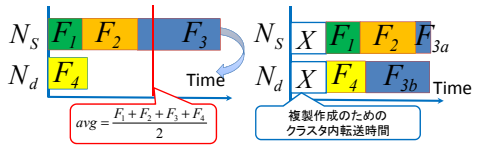
本節では「ファイルは分割してもよい」という仮定を取り除いた場合における転送複製の選択アルゴリズムと実行時複製作成アルゴリズムについて考察する．

また，18) の貪欲法複製選択アルゴリズムにおける，局所解に陥ることで性能が低下することがあるという問題を解決するために，貪欲法転送複製選択アルゴリズムにおける各ノードへのファイル初期割り当ての改善についても述べる．

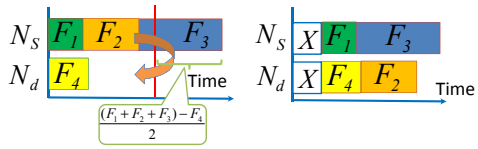
3.1 ファイル分割を許さない複製選択

ファイルの複数ノードへの分割割り当てを許さない場合，転送複製選択は組み合わせ問題となる．この問題は整数計画問題として定式化することで最適解を得ることが可能であるが，計算時間が長くなり並列複製作成のためのスケジューリングとして利用することは難しい．

そこで本研究では，はじめに先行アルゴリズムの線形計画法や貪欲法を用いて転送複製の選択を行う．この状態ではファイルが複数ノードへ分割割り当てされているため，この結果に対してファイルの分割割り当てを取り除く修正を行う．**図 4(a)** は図 2 の場合か



(a) 既アルゴリズムを用いた場合の複製作成



(b) 合計して $\frac{(F_1+F_2+F_3)-F_4}{2}$ の大きさに最も近いファイルサイズとなる複製作成対象ファイルを選択する

図 5 実行時複製作におけるファイル分割割り当て回避

ら $File_4$ と $File_5$ を除外した場合の転送複製の選択例である．この選択により転送時間は最短となるが，**図 4(b)** から分かるように $File_2$ が $Node_2$ と $Node_3$ に分割して割り当てられている．本研究で提案するファイル分割を許さない複製選択アルゴリズムでは，先行アルゴリズムによる複製選択の結果に対して，すべてのファイルを探索し，分割割り当てが行われていた場合は最も大きなファイル断片が割り当てられるノードに対して全ファイルを割り当てるように修正することを繰り返す．これにより最終的に**図 4(c)** のように分割割り当てのない状態とする．

3.2 ファイル分割を許さない実行時複製作成アルゴリズム

実行時複製作成アルゴリズムは**図 5(a)** のような基本モデルを用いて，割り当てられたファイルの総データ量に差がある 2 ノード間において割り当てられたファイルサイズの大きいノードから小さいノードへの複製作成を全ノードに対して適用し，全体の転送時間を効率的に減少させる．これは，クラスタ内外でのノード間データ転送において差がある仮定から，直接外部へ転送するよりも，一度内部で複製作成後並列転送を行ったほうが転送速度差の分高速であることを利用しており，転送元，転送先ノードにおいて割り当てられたファイルサイズが等しいときに転送時間最短となる．たとえば図 5(a)，**図 5(b)** では N_s から N_d に $\frac{(F_1+F_2+F_3)-F_4}{2}$ のサイズだけ複製を作成することで 2 ノード間においては転送時間最短となる．図 5(a) はファイルの分割を許している先行アルゴリズムを用いた場合の実行例である．基本モデルに基づき N_s に割り当てられているファイルのうちファイルサイズの大きいものから複製対象に選択していき，$\frac{(F_1+F_2+F_3)-F_4}{2}$ を超えた場合にはファイルを分割している．

一方，図 5(b) は本研究におけるファイル分割を許さない場合のアルゴリズム実行例である．ファイル分割を許さない本研究では効率的な転送時間削減のために転送元ノードから最適な複製作成の対象を選択しなければならず，図 5(b) の場合は F_1，F_2，F_3 から $\frac{(F_1+F_2+F_3)-F_4}{2}$ に最も近いファイルサイズとなるファイルの組み合わせを選び出すような組み合わせ最適化問題となる．この問題は NP 困難であり，最適解を求めるアルゴリズムはスケジューリングとして現実的ではない．そのため，本研究ではファイルサイズの大きい順にファイルを探索し，$\frac{(F_1+F_2+F_3)-F_4}{2}$ のようなしきい値を超えない範囲でしきい値に近づけていく貪欲的な解探索を行う．

3.3 ノードへのファイル初期配置決定アルゴリズムの改善

18) において転送複製選択で用いる貪欲法のアルゴリズムでは各ノードへのファイル割り当て初期状態を，全ノードを探索しあるファイルがそのノードで初めて出現したならばそのノードに割り当てるという単純な方法で決定していた．しかし，この方法ではファイルが特定のノードに対して偏って割り当てられているような初期配置となる場合があり，その後の解探索結果が悪化していた．

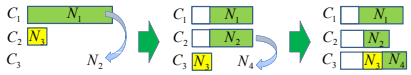
そこで，本研究のアルゴリズム改善ではまず各ファイルの複製数を第一優先，ファイルサイズを第二優先とした順位付けに従い，複製が少ない順に，ファイルサイズが大きい順にファイルをソートする．その後このファイル列をもとに各ノードに対する割り当てファイル数が均一になるようにファイル割り当て初期状態を決定する．複製数がファイルサイズよりも高い優先順位をもつ理由としては，複製を多く持つファイルを複製の少ないファイルの後で割り当てることにより，各ノードの割り当てファイルサイズの偏りを是正できる確率が高まると考えられるからである．

4. Rake によるファイル群転送

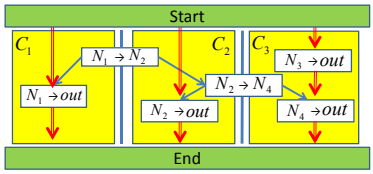
本節では Rake ビルドツール[4] について説明を行ったのち，Rake ツールを用いた提案並列ファイル複製作成のスケジュール記述と転送処理実行について述べる．

4.1 Rake

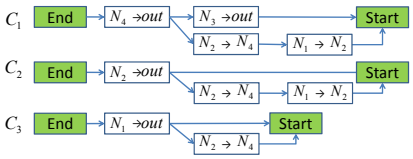
Rake とは Ruby[5] 内部 DSL として実装されているビルドツールであり，Make や Ant のようにソースコードのコンパイル記述などに汎用に用いること



(a) 実行時複製作成アルゴリズムの動作例



(b) 各ファイル複製作成処理の流れ例



(c) 各ファイル複製作成処理の依存関係例

図 6 各ファイル複製作成処理の流れと依存関係

ができる．文法は Makefile に似せて作られているが，Ruby 内部 DSL であることから基本的には Ruby の文法と同一であるため，Rake ファイルの記述や読解には Ruby の知識が必要になる．その反面，Ruby のすべての機能を Rake ビルドファイル内で用いることができるため，Makefile などに比べ自由度が高い．例えば，Makefile ではすべてのタスクやルールはあらかじめ記述されていなければならないが，Rake では Ruby スクリプトを利用して実行時にタスクを生成することが可能である．

4.2 ファイル転送スケジュールの Rake への適用

図 6(a) は提案アルゴリズムによって計算されたスケジューリングの一例である．はじめにノード N_1 とノード N_3 にファイルが割り当てられており，まず N_1 から N_2 に複製を作成し，次に N_2 から N_4 に複製を作成する．図 6(a) に示すように転送クラスタ内部で複製を作成することで外部への総転送時間を減少させることができる．

このときの，それぞれのファイル複製作成処理の流れは**図 6(b)** に示すようになる．コネクションに割り当てられているノードの基本的なファイル転送順序は図 6(b) 中の縦向きの 2 重矢印で示されている逐次的なものである．しかしながら，図 6(b) に示す通り，N_1，N_2，N_4 のように実行時複製作成を行うノードでは転送元クラスタ内複製作成処理とも依存関係がある．以上の複製作成の依存関係をグラフとして**図 6(c)** に示す．まず，提案アルゴリズムでは，送信元クラスタ内においてすべてのノードは他ノードへファイルを送信したのち，他のノードからファイルを受信することはないため，このグラフは循環することはない．そのため，提案アルゴリズムで計算されたファイル複製作成のスケジュールは Makefile のような記述によって宣言的に書き下すことができる．また，このタスク記述は Rake を利用することによりスケジューラの出力をもとに Ruby スクリプトによって動的に作成することができる．

5. 評　　価

5.1 設　　定

図 1 のように Site A の各ノードに分散して配置されたファイル群をすべて Site B の各ノードへ転送する場面を想定し，すべてのファイル複製を作成するための転送時間とスケジューリング計算時間をシミュレーションにより評価する．作成したシミュレータは以下のパラメータをとり，一様ランダムに各ファイルサイズ，ノードに対するファイル配置を決定する．また，線形計画問題のソルバとしては GLPK(GNU Linear Programming Kit) を用いた[11]．

Number of Files　ファイルの数．

Number of Nodes　割り当てられるノードの数．

Number of Connections　コネクションの数．

Repl　ファイル複製の存在しやすさ．パラメータ $0 \leq Repl \leq 0.99$ により，あるファイルが $N(>1)$ 個のノード上に存在する確率を

$$(1 - Repl)Repl^{N-1} \quad (1)$$

として定義する．

Bias　ファイルのノードに対する偏りやすさ．パラメータ $0 \leq Bias \leq 0.99$ により，N 個のノード中 n 番目のノードにファイルが存在する確率を

$$\frac{Bias(1 - Bias)^{n-1}}{1 - (1 - Bias)^N} \quad (2)$$

として定義する．また，$Bias = 0$ の場合一様ランダムに決定する．

ファイル複製の存在しやすさのパラメータ $Repl$ について，$Repl^N$ は N 個以上のノードにファイル複製が存在する確率を示している．さらに，式 (1) は公比 $Repl < 1$ の等比数列であるため，ファイル全体に対する複製を N ノードに持つファイルの割合は，複製数 N に対して指数的に減少していくことを表す．すなわち，$Repl$ を小さくすればほとんどのファイルが

複製を持たないことを，$Repl$ を大きくすれば複製を多く持つファイルが多くなる．

また，ファイルのノードに対する偏りやすさのパラメータ $Bias$ の設定について，式 (2) は初項 $\frac{Bias}{1-(1-Bias)^M}$，公比 $(1-Bias)$ の等比数列であることから，$Bias$ を小さく設定することで初項が小さく，かつ公比が 1 に近くなり，すべてのノードに対して均一なファイル配置となる．逆に $Bias$ を大きく設定することで特定のノードに偏ったファイル配置となる．

以上のパラメータを変化させファイル転送所要時間に与える影響を調べる．上記で述べた以外のパラメータについて**表 1** のように設定する．また，複製数に関するパラメータ $Repl$ とファイル配置に関するパラメータ $Bias$ については，複製がほぼ無い場合と十分に有る場合，極端に特定のノードに偏ってファイルが分布している場合と多くのノードに均等に分散配置している場合の組み合わせ 4 パターンについて，それぞれのパターンにおけるシミュレーション結果の傾向から複製数とファイル分布が提案アルゴリズムの性能に与える影響を調べる．

$Repl$ については $Repl = 0.01$（ファイル複製が少ない場合）と $Repl = 0.7$（複製が多い場合），$Bias$ については $Bias = 0.01$（ファイル配置の偏りが小さい場合）と $Bias = 0.99$（偏りが大きい場合）と設定した．以降の考察では $Bias = 0.99$ の場合は “bias”，$Bias = 0.01$ の場合は “nbias”，同様に $Repl = 0.7$ の場合は “repl”，$Repl = 0.01$ の場合は “nrepl” と表現する．たとえば，“bias_nrepl” ならば $Bias = 0.99$ かつ $Repl = 0.01$ を示す．本論文で設定した，$Repl = 0.01, 0.7$，$Bias = 0.01, 0.99$ における複製数，ファイル配置を**図 7**，**図 8** に示す．また，過剰に多くのファイル複製が存在した場合，組み合わせ問題の最適解が多く存在することとなり，問題として簡単になると考えられる．そのため，複製が過剰にならずに，十分に複製が存在する $Repl = 0.7$ と設定している．図 7，図 8 より，$Repl = 0.01, 0.7$，$Bias = 0.01, 0.99$ とすることで複製の十分な場合とほとんどない場合，ファイル配置の均一な場合と偏っている場合についてよく再現できていると考えられる．

5.2 ファイル分割を許さない複製選択の評価

転送複製選択時において分割割り当てを許さない場合のスケジューリング結果に対する影響について評価する．貪欲法と線形計画法のそれぞれのアルゴリズムについて本研究で提案した分割割り当てを避ける修正を施す場合と分割を許す場合の 2 パターンについて，ファイル数を変化させながら全複製作成に要するファイル転送時間の変化をシミュレーションにより評価する．**図 9**，**図 10** はそれぞれ貪欲法，線形計画法について分割割り当てを許す場合の転送時間に対して分割割り当てを許さない場合の転送時間の比をとったグラフである．1 よりも大きければ，分割割り当てを禁止することで，より転送時間が伸びていることを表す．図 9，図 10 よりまず，ファイル配置の偏りが大きい場合についてはどちらの場合においても ±0.02 程度の比に収まっている．しかし，ファイルの偏りが小さい場合には転送ファイル数が少ない場合において転送時間比が大きくなっている．これは，貪欲法の場合はファイル分割の修正数がファイル数によらずにほぼ一定となっているためであると考えられる．貪欲法のアルゴリズムは各ノードをコネクションにスケジューリングしたのち，各コネクションの差が小さくなるようにファイルの再配置を行っていくが，ファイル再配置においてそのコネクションから最後に再配置されるファイルが分割される可能性がある．そのためこの時のファイル

表 1 ファイルサイズとネットワークバンド幅設定

最大ファイルサイズ	12.5 GB
平均ファイルサイズ	6.25 GB
送信元クラスタ内バンド幅	1 Gbps
PC クラスタ間バンド幅	10 Gbps
トラフィックシェーピング後のバンド幅	200 Mbps
基準となる送信元ディスク I/O	400 Mbps

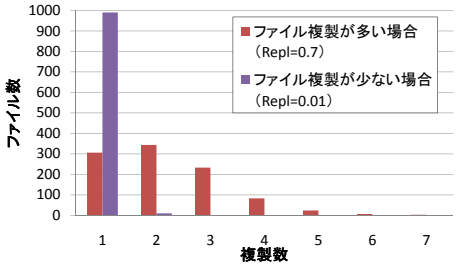


図 7 複製数当たりのファイル数．ノード数は 100

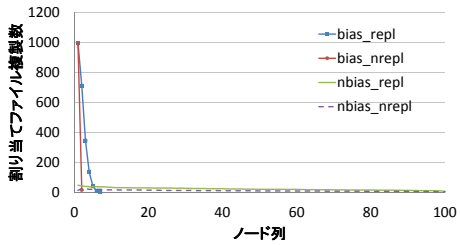


図 8 各ノードへのファイル割り当てとファイルの偏りの一例，複製数の関係.100 ノードに対し 1000 個のファイルをシミュレータにより割り当てる．すべての複製を含むためファイルの総和は 1000 を超える．

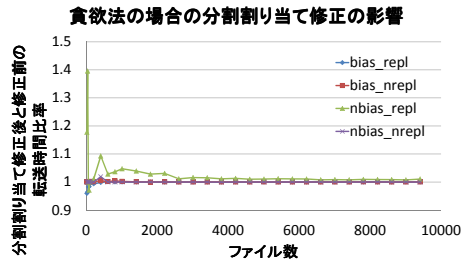


図 9 ファイル数変化に伴うファイル分割を許さない貪欲法による複製選択の転送時間への影響

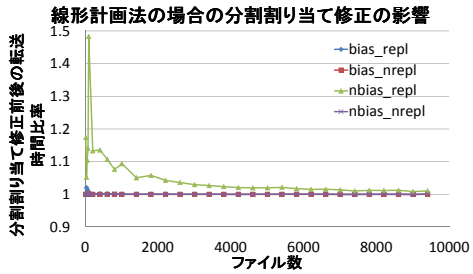


図 10 ファイル数変化に伴うファイル分割を許さない線形計画法による複製選択の転送時間への影響

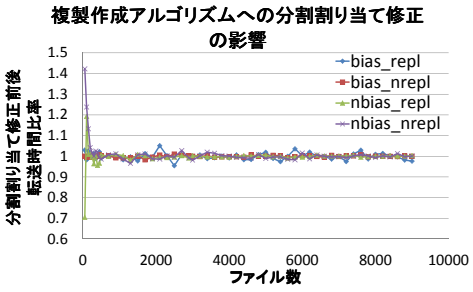


図 11 ファイル数変化に伴うファイル分割を許さない実行時複製作成処理の転送時間への影響

分割の修正数はコネクション数によると考えられ，転送ファイル数が少ない場合には総転送時間が小さくなり，分割割り当て修正の影響が大きくなる．

一方，線形計画法の場合は各ノードの割り当てファイルサイズを平均的にするために，各ファイルごとに分割される可能性がある．そのため，分割の修正数が多くなることが考えられるが，コネクション当たりの分割修正数も増えるため，ファイル数が増えれば特定のコネクションだけにファイルが偏り転送時間が増加する確率も低下すると考えれる．そのため，ファイル数を増やすことにより転送時間比は 1 に収束していく．

5.3 ファイル分割を許さない実行時複製作成アルゴリズムの評価

次に実行時複製作成においてファイル分割割り当てを許さない場合について評価する．前節と同様に実行時複製作成のアルゴリズムに対してファイル分割割り当てを許す場合と本研究で議論する許さない場合について，シミュレーションにおいて転送ファイル数を増加させた場合の各予想転送時間の比について調べる．転送ファイル複製の選択には貪欲法を用いている．その結果を**図 11** に示す．ファイル数が少ない場合において，ファイル分割割り当て禁止の影響により転送時間に差が出ている．これも前節と同様にファイル数自体が少ないために，総転送時間が短いため分割割り当て禁止の影響が強く出ているためであるといえる．実行時複製作成の場合もファイル数増加に伴い次第に転送時間比は 1 に収束していく．これより，ファイル数増加に伴う転送時間増加に比べ，提案アルゴリズムにおけるファイル分割割り当て禁止の影響は小さいといえる．

表 2 マシンスペック

CPU	Intel(R) Xeon(R) CPU 3060 2.40GHz
memory	6GB
OS	Linux 2.6.18-194.11.4.el5

また，評価に用いたマシンスペックを**表 2** に示す．この条件のもと最もアルゴリズムに計算に時間のかかる “bias_repl” のパターンにおいてファイル分割割り当てを許す場合の計算時間はファイル数 10,000 個のとき 40 秒程度であった．このとき，提案する分割を許さないアルゴリズムの計算時間は上記の時間とミリ秒単位の違いであった．全体を通しても分割を許さない場合の計算時間と分割を許す場合の計算時間の値の比は 1.02 以下に収まっていた．本節におけるシミュレーションではファイル分割割り当てを許さない貪欲法を用いた転送複製選択と実行時複製作成アルゴリズムの双方の合計計算時間を評価していることから，この結果から提案アルゴリズムは十分に高速であるといえる．

5.4 貪欲法アルゴリズムにおけるファイル初期配置改善の評価

図 3.3 節で述べた貪欲法転送ファイル複製選択アルゴリズムのための各ノードへの初期ファイル割り当て改善アルゴリズムを評価するために，先行アルゴリズムの単純な初期割り当て決定方法と提案アルゴリズムを比較評価する．以下図内において，先行アルゴリズムでの単純な決定方法を “simple”，提案アルゴリズムを “finfo” と表現する．

図 12 はファイル数を変化させた場合において，“simple” アルゴリズムと提案アルゴリズムをそれぞれ利用した貪欲法による転送複製選択の結果からすべての

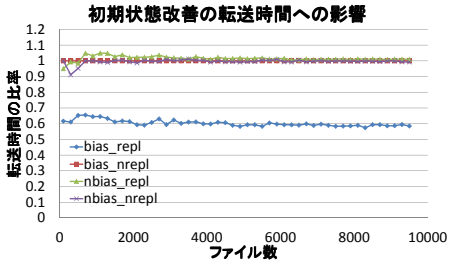


図 12 ファイル数変化に伴う初期割り当て改善の転送時間への影響

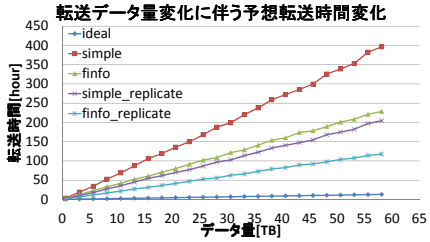


図 13 “bias_repl” の場合の転送データ量変化に伴う転送時間の変化

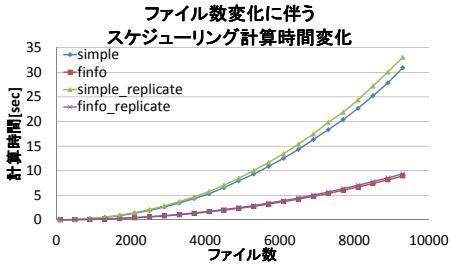


図 14 “bias_repl” の場合のファイル数変化に伴うスケジューリング計算時間の変化

ファイル複製に要する総転送時間を求め，その比の変化をグラフとしている．値が 1 よりも小さい場合は提案アルゴリズムを用いることにより従来方法を用いるよりも転送時間を短縮化していることを示す．図 12 より，ファイルが特定のノードに偏って配置され．複製が多く存在している場合（“bias_repl”）において約 40%高速化していることが分かる．これは，ファイル配置の偏りが大きく複製が多く存在する場合，既アルゴリズムでは特定のノードに偏ったファイル配置となりやすく，そのために貪欲法による複製選択では局所解に陥いることで著しく性能が悪化していたためである．しかしながら，提案アルゴリズムでは初期配置決定時点において均一なファイル配置を行うことで局所解を回避していると考えられる．

次にファイル配置の偏りが大きく複製が多く存在する場合（“‘bias_repl’）の場合においてファイル数を変化させた時のシミュレーションによる転送時間とアルゴリズムの計算時間を**図 13**，**図 14** に示す．図 13 においては，1 ファイルの期待値が 6.25GB となることから，ファイル数から転送データ量の期待値を求め，横軸としている．また，図中において “simple”，“finfo” がそれぞれ既アルゴリズムと提案アルゴリズムを用いた貪欲法複製選択の結果であり，さらに “simple_replicate”，“finfo_replicate” がそれぞれ同様に貪欲法で複製選択を行ったうえで，実行時複製作成までを行った結果である．以上，すべての場合においてファイル分割は許していない．まず図 13 より，図 12 と同様に初期配置の改善により複製作成を行わない場合，行う場合両方とも既アルゴリズムの結果に比べ約 40%高速化している．これは，線形計画法を用いて複製選択を行った場合と同等の性能であった．

また，図 14 より計算時間は初期割り当ての改善により，先行アルゴリズムを用いた場合の約 1/3 となった．これは，より良い初期割り当てにより，貪欲法による解の探索空間を小さくすることができたためである．“bias_repl” だけでなく他のパターンにおいても同様の理由により同等程度以上の高速化が実現できている．

6. 関 連 研 究

広域環境において信頼性のあるファイル転送には RFT[10] がよく用いられる．RFT は GridFTP[12] の上位に位置し，再送などの信頼性をもつファイル転送を提供している．一方，GridFTP は広帯域・高遅延ネットワークを想定し，転送性能を保証するためにマルチストリーム転送などの機能を持つ．これら従来のファイル転送は単一ファイルを効率的に転送することが目的であり，本研究で想定する複数ノードに分散した多ファイルの効率的転送を行うためには個々の転送最適化だけでなく全体スケジューリングが必要となる．複数ノードからの転送に関する関連研究としては CEP[15] が挙げられる CEP は N ノードから M ノードでファイル転送を行うが，効率的転送のために中央スケジューラにより転送ファイル，宛先，送り先のスケジューリングを行う．CEP はこれら本研究との類似点が多いが，本研究が広域での大規模データの複製作成を目的とし，輻輳の対策などを行っている点が大きく異なる．

また，本研究ではスケジュールを非循環グラフとして表現し，実行する．このように依存関係のあるタスク群をグラフとして表現することはワークフローに関する研究でよく行われる．GXP Make[14] は Makefile

を用いて，Pwrake[13] は Rake を用いてワークフローを記述している．また，DAGMan[2]，Pegasus[7] は非循環グラフを用いて計算タスクとデータステージングのためのデータ転送タスクを組み合わせた全体ワークフローを記述している．

7. まとめと今後の課題

PC クラスタ間複製作成を全体として効率的に行うために，PC クラスタ各計算ノード間の複製作成処理をスケジュールするスケジューリングアルゴリズムの提案を先行研究として文献[18] において行った．しかしながら，PC クラスタ間複製作成スケジューリングの問題は NP 困難であり，簡単化のためにファイル分割を許していた．そのため，実際のファイル複製に先行アルゴリズムを用いた場合，ファイルが受信クラスタ内で複数ノードに分割して格納される問題があった．そこで，本論文ではファイル分割を許さない転送ファイル複製の選択アルゴリズムと実行時複製作成アルゴリズムについて提案・評価し，その結果を Rake ツールによって実行する手法について述べた．また，先行アルゴリズムの問題点の一つであった貪欲法複製作成アルゴリズムが局所解に陥ることによってスケジューリング性能が悪化する問題の解決のためにファイル初期配置決定アルゴリズムの改善手法を提案し，評価した．

シミュレーション評価の結果，ファイルの分割を許さない提案アルゴリズムを用いた場合においても，ファイル分割を許す先行アルゴリズムを用いた場合と比べ転送ファイル数が少ない場合を除いて総転送時間の差は 5%以内程度であった．計算時間の差も 2%以内であり，提案アルゴリズムは十分高速で実用的であるといえる．また，初期割り当て改善アルゴリズムによって既アルゴリズムを用いた場合に比べ転送時間を最高 40%短縮化することができた．さらに，計算時間に関しても既アルゴリズムを用いた場合に比べ最高約 1/3 程度まで高速化ができた．

今後は本論文によって提案したスケジューリングを用いて，PC クラスタ間のファイル複製作成処理の自動化を進める．その後，実環境において測定評価を行っていく．

謝辞 本研究の一部は，文部科学省科学研究費補助金特定領域研究課題番号 21013005 および文部科学省次世代 IT 基盤構築のための研究開発「e-サイエンス実現のためのシステム統合・連携ソフトウェアの研究開発」，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) Avian flu grid. `http://avianflugrid.pragma-grid.net/`.
2) DAGMan: The directed acyclic graph manager. `http://www.cs.wisc.edu/condor/dagman/`.
3) ILDG. `http://ildg.sasr.edu.au/Plone`.
4) Rake, `http://rake.rubyforge.org/`.
5) Ruby, `http://www.ruby-lang.org/`.
6) Krishna P. Belkhale and Prithviraj Banerjee. An Approximate Algorithm for the Partitionable Independent Task Scheduling Problem. In *Proceedings of ICPP, Vol. 1*, pp. 72–75, 1990.
7) E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.H. Su, K. Vahi, and M. Livny. Pegasus: Mapping scientific workflows onto the grid. In *Proceedings of AxGrids '04*, pp. 11–26, 2004.
8) Flavia Donno and Maarten Litmaath. Data management in WLCG and EGEE. Worldwide LHC Computing Grid. Technical Report CERN-IT-Note-2008-002, CERN, Geneva, Feb 2008.
9) R. L. Graham and R. L. Grahamt. Bounds on Multiprocessing Timing Anomalies. *SIAM Journal on Applied Mathematics*, Vol. 17, Issue 2, pp. 416–429, 1969.
10) R.K. Madduri, C.S. Hood, and W.E. Allcock. Reliable file transfer in grid environments. In *Proceedings of LCN '02*, p. 0737, 2002.
11) A. Makhorin. GLPK (GNU linear programming kit). `http://www.gnu.org/software/glpk/glpk.html`.
12) I. Mandrichenko, W. Allcock, and T. Perelmutov. GridFTP v2 protocol description.
13) M. Tanaka and O. Tatebe. Pwrake: a parallel and distributed flexible workflow management tool for wide-area data intensive computing. In *Proceedings of HPDC '10*, pp. 356–359, 2010.
14) K. Taura. GXP: An interactive shell for the grid environment. In *Proceedings of IWIA '04*, pp. 59–67, 2004.
15) E. Weigle and A. A. Chien. The Composite Endpoint Protocol (CEP): scalable endpoints for terabit flows. In *Proceedings of CCGRID '05*, pp. 1126–1134, 2005.
16) 高野了成, 工藤知宏, 児玉祐悦. 精密な帯域共有とトラフィック隔離を実現するパケットスケジューリング方式. 情報処理学会研究報告，2009-HPC-119(9)，pp.67-72, 2009.
17) 須田礼仁. ヘテロ並列計算環境のためのタスクスケジューリング手法のサーベイ. 情報処理学会論文誌. コンピューティングシステム, Vol. 47,

No. 18, pp. 92–114, 2006.
18) 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. Vol. 3 No. 3, pp. 113–125, 2010.