社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－ARC－182 (9)
2009－HPC－119 (9)
2009／2／26

並列ファイル転送のためのスケジューリングアルゴリズム

鈴木克典[†1] 建部修見[†2]

本稿は我々が想定する並列ファイル転送システムにおける，ファイル転送タスクのスケジューリングアルゴリズムに関する提案である．想定システムはグリッド環境においてクラスタ間でファイル転送を行うものであり，各ノードに複数の複製が存在することを仮定する．このとき適切な複製選択，転送順序の決定，複製の動的作成を行うことで，最適な転送時間を求める．我々は，この問題を定式化し，リストアルゴリズムを基本とした手法として実装した．提案手法を評価した結果，特定のノードにのみにファイルが偏って分布している場合でも予想転送時間を短縮できることを確認した．

The Task Scheduling Algorithm for Parallel File Transfer System

Katsunori Suzuki[†2] and Osamu Tatebe[†2]

We present a task scheduling algorithm of the parallel file transfer system. It is asuumed that the system does file transfer to and from clusters in grid environment, and two or more replicas exist in each cluster. In this situation, to optimize the transfer time, proper transfer scheduling including replica selection and dynamic replica creation should be investigated. We build a model to solve the problem and implement algorithms based on the list-algorithm. Performance evaluation shows that the proposed replica selection algorithm and the replica creation algorithm provide better result than a simple list scheduling in unevenly file distributed case.

1. はじめに

近年，広域で巨大なデータを処理する要求が高まっている．科学研究分野では実験によりデータを生成し，その分析を必要とする．それらは実験設備，データを格納するためのストレージや分析に要する処理能力などの理由から距離的に離れた場所で行われることが多い．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世界規模のグリッドの利用が計画されている．

巨大なデータを効率的に転送するためにはバンド幅を有効に使用しなくてはならない．現在，グリッドにおいてはファイル転送のために FTP を拡張した gridFTP[1]，RFT[2] や FTS[3] などが用いられている．

これらは，1 ノード対 1 ノードの転送を目的としたものである．しかし，巨大なデータを 1 ノード対 1 ノードで転送することは効率が悪い．また，スループットを増やそうと多ノード対多ノードで転送を行うと衝突により逆効果となる．そこで我々は，多対多のデータ転送を効率的に行うためのシステムを考えている．提案システムはクラスタ間のデータ転送を行う．データはクラスタに分散配置されており，基本的に各ノードが転送を行うが，システムにより転送量，転送順序やファイルの割り当ては管理される．

本稿では，提案システムのためのファイル転送タスクのスケジューリングアルゴリズムに関する考察を行う．さらにシステムの特徴を踏まえ，問題をモデル化し，アルゴリズムを設計する．

本稿の構成について述べる．まず次節で提案システムの概要を説明し，スケジューリングアルゴリズムが満たすべき要求を分析する．さらに，その要求をまとめ問題設定を行う．3 節では前節で設定した問題を定式化し，4 節において問題を解くための提案アルゴリズムを説明する．5 節では提案アルゴリズムを評価し，考察を行う．最後にまとめと今後の課題について考察する．

2. 並列ファイル転送システムにおける問題の設定

2.1 提案する並列ファイル転送の概要

提案する並列ファイル転送システムはクラスタ間でのファイル転送のための機構である．ファイルを高速に転送しようと試みた場合，ディスクアクセスがボトルネックとなることが多い．そこで，複数ファイルまたは巨大ファイルを効率よく転送するために，各ノードへファイル転送タスクを振り分け，複数ノードが割

†1 筑波大学第三学群情報学類
College of Information Science, Third Clusters of Colleages Univercity of Tsukuba
†2 筑波大学大学院システム工学研究科
Graduate School of System and Infomation Engineering of Tsukuba

り当てられたファイル転送タスクに従い並列に転送処理を行う．しかし，各ノードが自由に転送を行った場合，ネットワーク上でパケットが競合しかえって速度低下を起こしてしまう．そのため各ノード上で転送速度のコントロールを行う．

各ファイルはブロックに分割され複数ノードにより同時に転送される．特定のノードにファイルが偏っていた場合やファイルが極端に大きい場合，特定のノードに負荷が偏りボトルネックとなってしまう．そこで転送処理を行っていないノードへ動的に複製を作成することで，負荷の分散を図る．

2.2 クラスタ間並列ファイル転送システムのタスクスケジューリングアルゴリズムに対する要求

2.2.1 同時ディスクアクセスによる速度低下

一般的に一つのハードディスクに対し同時に複数ファイルを読み書きした場合，シークが頻発しディスクアクセス時間が極端に長くなる．そのため一つのノードが同時に複数のファイル転送をおこなうようなタスク割り当ては避けなければならない．

また，外部への転送中にクラスタ内で動的に複製を作成する際にも配慮が必要である．これはファイル転送タスクを実行中のノードに，複製を作成する場合であり，一つのノードが同時に外部へのファイル転送タスクとクラスタ内でのファイル複製作成タスクを実行する事態は避けるべきである．

2.2.2 クラスタ内ノードに対するファイル配置の不均一さ

提案システムはクラスタ内の各ノードに分散して格納されている複数のファイルを並列に転送することを想定している．しかしながら，ある特定のノードにのみ，ファイルが偏在している場合も考えられる．この場合ファイルが存在するノードからの転送時間が全体のボトルネックとなってしまう．

そのため，提案システムでは外部への転送タスクを実行中に，クラスタ内で，ファイル転送タスクを実行していないノードへファイルの複製を動的に作成する機構を考えている．これにより全体のタスク実行時間が短縮できる．タスクスケジューラは前節の制限を踏まえ外部への転送タスクとクラスタ内でのファイル複製作成タスクを，同一ノードに，同時に，複数個割り当ててはならない．

2.2.3 ファイル複製の存在

複数のノードに，ファイルの複製が複数個存在するということが考えられる．この場合タスクスケジューラは適切なファイルを選択することによりノードに対する負荷を分散させることでき，全体としての転送速度向上が可能である．

2.2.4 使用可能バンド幅による制約

提案システムでは，タスクスケジューラはディスクアクセス速度と使用可能バンド幅から適切なバンド幅を各ノードに設定し，その転送速度から予想実行時間を予測した上で適切なファイル転送タスクを各ノードに割り当てなければならない．

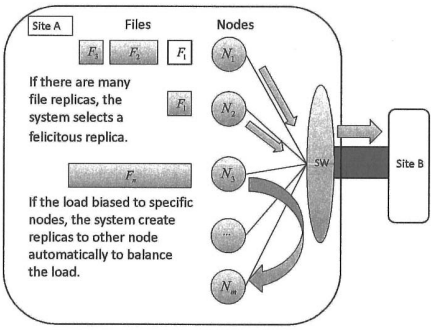


図 1 提案並列ファイル転送システムの概要

2.3 問題設定のまとめと形式化

前節までの分析からスケジューリングアルゴリズムへの問題要求をまとめる．

(1) 複数のノードのストレージに複数のファイルが分散して格納されている状況下で効率的なファイル転送順序を求める

(2) 使用可能なバンド幅には制限がある．データパケットの衝突による速度低下を避け，バンド幅を効率的に活用するために各ノードの転送速度，転送順序をスケジューリングする

(3) クラスタ内にファイルの複製が複数存在している場合，全体として転送時間が短くなるよう適切なノード上のファイルを選択する

(4) 特定のノードにファイルが偏っている場合，負荷を分散させるためにクラスタ内で動的に複製を作成する．

(5) ディスク同時アクセスによるデータ読み出し，書き出し速度の低下を避けるために特定のノードにおいて同時に複数のファイル転送，ファイル複製作成を行ってはならない

3. 定 式 化

2.3 で考察した問題を定式化する．ここで，ファイル集合を F，ノード集合を N と表し，ファイル数を n, ファイル名を $\{f_i|1 \leq i \leq n\}$, ノード数を m, ノード名を $\{N_i|1 \leq i \leq m\}$ とする．

3.1 コネクション概念の導入

クラスタ間バンド幅を抽象化した「コネクション C」を導入する．このとき仮定としてコネクション C は $\{c_i|1 \leq i \leq k\}$ に分割することが可能であり，その総量は一定 $(C = \sum_{i=1}^{k} c_i)$ であるとする．これは，各ノード間の転送パケットが競合し速度の低下が起きないよう，よく帯域制御されていることを仮定している．

ここで考えるコネクションとは転送処理に使用可能なバンド幅を抽象化したものである．つまり，ノード i にコネクション c を割り当てることは，ノード i に c を割り当てている時間中 c のもつバンド幅で転送処

理を行う許可を与えることを意味する．

また，本モデルでは簡単化のためにコネクションは固定サイズで分割されるものとする．

ノード i 上のファイルを外部へ転送する場合，ディスク I/O 速度やノード i からスイッチ sw 間のネットワーク速度など一連の転送処理のボトルネックを，ノード i の I/O 速度 v_i と置く．

ノードに割り当てられるコネクションのバンド幅は $\{v_i|i=1,\cdots,m\}$ をもとに求める．クラスタ間のバンド幅を B と表したとき，コネクションのバンド幅は，

$$v(c)=\min_{i=1,\cdots,m}(v_i,B)$$

とする．

3.2 モデル化

3.2.1 問題の簡単化

コネクション概念の導入により，我々は本問題をコネクション集合 C に対するファイル集合 F のスケジューリング問題としてモデル化した．このとき，注意すべき問題として，ノードに対する同時ディスクアクセスの制限がある．この制限により，ファイルをコネクションに割り当てるにあたり，どのノード上に存在するファイルであるか，を考慮しなければならず，難しい問題となる．

そこで，我々は「ファイルは独立かつ分割可能である」の条件を利用し，同時ディスクアクセスの制限を回避した．ノード N_i に格納されているファイル集合 F_i は，分割可能なファイルの集合であるため，一つのデータまとまりであるとして扱うことができる．また，ファイル集合 F_i は互いに独立なファイルの集合であるため，F_i を外部へ転送するタスクを T_i と表したとき，ノード N_j 上のファイル集合 F_j の転送タスク T_j は，T_i と互いに独立なタスクであるといえる．

以上のように，ノード上の複数のファイルを一つのデータであると考えることにより，求めるコネクション集合 C に対するファイル集合 F のスケジューリング問題は，コネクション集合 C に対するノード集合 N の独立タスクスケジューリング問題に帰着する．

3.2.2 ファイル複製の作成

ここで，ファイルを分割し，負荷の偏っているノードから，別のノードに複製を作成することを考える．これは他のノードが外部への転送中にクラスタ内で裏で行うものであり，スケジューリングに際して同時ディスクアクセスの制限を考慮しなくてはならない．

変数としてある時間 t を与えたとき，その時間中にコネクション c_k に割り当てられているノードを $N_{(t,c_k)}$ とする．このとき，$N_{(t,c_k)}$ と，複製作成元ノード N_s，複製作成先ノード N_d はすべて異ならなければならない．この条件は，以下のように表わされる，

$$\exists t.\forall i,j.\ ((1\leq i<j\leq k)\wedge (N_{(t,c_i)}\neq N_{(t,c_j)}\neq N_s\neq N_d))$$

3.2.3 コネクション当たりの予想転送時間の計算

ここで，ノード i が c_k に割り当てられているかどうかを $(x(i,c_k)\in X|X=\{0,1\})$ とする．$x(i,c_k)=0$ のときノード i はコネクション c_k に割り当てられておらず，$x(i,c_k)=1$ のときファイル i はコネクション c_k に割り当てられていることを表す．

求めるコネクション当たりの転送実行時間は，割り当てられたノード i のファイル転送タスクの転送時間 l_i の総和である．このときファイルの総転送時間 L_{c_k} は，

$$L_{c_k}=\sum_{a=1}^{m}(l_a\times x(a,c_k))$$

と表せる．

4. 提案アルゴリズムの説明

4.1 基本戦略

提案するアルゴリズムは大まかな割り当てを決定する前処理，さらに前処理の結果をもとに複製選択を行い，最後に複製作成行う，三段階のスケジューリングである．

3.2 で述べたように解くべき問題は複製選択，複製作成を含めた独立タスクスケジューリングである．一般的に互いに独立なタスクのスケジューリング問題は NP 困難であり実時間で最適解を求めることは難しく，リストアルゴリズムなどのよい近似解を与えるアルゴリズムが用いられる．提案アルゴリズムでも前処理にリストアルゴリズムを利用したスケジューリングを行う．

ここで，リストアルゴリズムについて簡単に説明する．リストアルゴリズムはタスク列 $\{t_i|1\leq i\leq n\}$ とそれを処理するマシン $\{M_j|1\leq j\leq m\}$ が与えられたとき，マシン集合に対するタスク列のスケジューリングを計算する．手順としては，タスク列の先頭タスクから順にタスクを取り出し，その時点でタスクを割り当てた時の終了時刻が最も早いマシンを選択し割り当てを決定していく手法である．[4)]

4.2 前処理

ノード集合 N をコネクション集合 C にたいしてスケジューリングするためには，まず転送するファイル複製を選択し，どのノードのファイルを転送するかを決定しなければならない．提案アルゴリズムではこの段階での複製の選択を負荷分散，転送時間短縮化を考慮せずに決定する．さらに，ファイル複製を決定したノードに対して，それを降順に整列したリストアルゴリズムを適用する．この結果によりどのファイルの複製を選択し直せばよいのかという情報を得ることができる．

4.3 複製選択

前処理後，コネクションに対して負荷が偏っている場合，複製の選択を行い負荷を分散させ，全体の予想転送時間の短縮を考える．

図 2 を用い簡単に説明する．

(1) ノード集合 N をリストアルゴリズムを用いてコネクション集合 C にスケジューリングし，ど

```
1: N ← {n_1, ···, n_m} –(1)–
2: C ← {c_1, ···, c_l}
3: schedule N to C by list_algorithm
4:
5: Define L_c as prepared finish time of connection c
6: margin ← permitted error / 2
7: avg ← avg(L_c|c ∈ C)
8: upper_bound ← avg + margin
9: lower_bound ← max(avg − margin, 0)
10: C_overs ← φ
11: C_unders ← φ
12: –(2)–
13: for all c ∈ C do
14:     if L_c > upper_bound then
15:         C_overs ← {c} ∪ C_overs
16:     end if
17:     if L_c < lower_bound then
18:         C_unders ← {c} ∪ C_unders
19:     end if
20: end for
21: sort C_overs bigger.
22: sort C_unders smaller.
23: –(3)–
24: for all c_o ∈ C_overs do
25:     while L_co > upper_bound do
26:         if Is all task gotten assiged for c_o? then
27:             break
28:         else
29:             get a task t from c_o
30:         end if
31:         for c_u ∈ C_unders do
32:             if c_u has replica of t then
33:                 break
34:             end if
35:         end for
36:         –(4)–
37:         if the replica of t is found then
38:             size = size_of t
39:             if L_co − lower_bound < size then
40:                 size = L_co − lower_bound
41:             end if
42:             if upper_bound − L_cu < size then
43:                 size = upper_bound − L_cu
44:             end if
45:             Reallocate size of t to c_u from c_o
46:             Sort C_unders smaller.
47:         end if
48:     end while
49: end for
```

図 2 複製選択のアルゴリズム select_replica

の程度まで偏りができるのか計算する．平均化の目的となる値 avg と許される誤差範囲を設定する．

(2) 負荷の偏るコネクションを探索する．$upper\_bound < c_{over}$ と $lower\_bound > c_{under}$ を見つけ，以後，c_{over} に割り当てられているファイルを c_{under} に振り分けていくことを考えていく．

(3) C_{overs} のすべての要素に対し，(3) と (4) を行う．$c_o \leq upper\_bound$ または $c_o = \phi$ となるまで，タスク t を取り出し，t が C_{unders} に割り当てられているノードの中にレプリカを持つかどうかを探索する．C_{unders} に対してはサイズに関し昇順に，C_{overs} に関しては降順に探索を行いできるだけ仕事量を減らす．

(4) (3) により，レプリカが見つかったとき振り分けるファイルサイズを決定する．もしも，$c_{over} < lower\_bound$ となってしまう場合，複製を作成したために c_{over} が新しく c_{under} となってしまう．逆に $c_{under} > upper\_bound$ となってしまう場合，c_{under} が新しく c_{over} となってしまう．これを許すとアルゴリズムが停止しなくなるため，$c_{over} \geq lower\_bound$ かつ $c_{under} \leq upper\_bound$ となるサイズを計算し複製の振り分けを行う．

4.4 複製作成

複製選択を行ったとしても負荷の偏りが解消できない場合，ノード間における動的な複製作成を行う．このときできる限り外部への転送に影響を与えないように考慮する．基本的には以下の手順により実現する．

(1) 変数として時間 $time, t_0, t_1$ を導入し，初期値として $time = t_0 = 0$ とする．

(2) 負荷が偏っており全体のボトルネックとなっているノードを N_s として抽出する．また残りのノードをリストアルゴリズムによりコネクションに割り当てる．

以下，N_s のノード全てにおいて以下の手順を行い，N_s から他ノードへ複製を作成する．

(3) $t_0 = t_0 + time$ とし，時間 $time$ において転送を行っているノードを N_t とする．N_t の予想転送終了時間が t_0 に最も近いものを t_1 とする．このとき $t_1 = 0$ であり，全てのコネクション負荷が平均化されているならば (9) へ．

(4) $time = t_1 - t_0$ とし時間 $time$ の間で複製を作成できるノードと複製作成量を計算する．

(5) 複製作成先ノードを集合 $N - N_t - N_s$ から求め，複製元ノードを集合 N_s から求める．

(6) 複製量はクラスタ内転送速度 v_in と外部へのコネクション c より計算する．クラスタ内部と外部との速度比を v_{in} と c から計算し，時間 $time$ の間に転送できるデータサイズを計算する．

(7) (5),(6) で計算した値をもとに，複製作成処理の割り当てを $max(N_s, N - N_d - N_t)$ の数だけ繰り返す．終了後，$N_s = \phi$ ならばアルゴリズム終了，そうでないならば (8) へ．これはコネクションが平均化し収束している状態．

以上の処理は，タスクの終了時間を一つの区切りとし，この差分をとりながら複製転送元ノード，複製先ノード，複製作成量を計算している．これにより，他ノードが外部へ転送中に複製を作成するスケジュールが立てられ，そのオーバーヘッドも隠ぺいすることが可能である．

しかしながら，コネクション数が増加するほど，ループを繰り返すほど差分は 0 に収束していく．そのため，以上の処理が適用できなくなり，どうしても複製作成のオーバーヘッドが隠せなくなる．このとき，できる限り複製作成の影響を抑えるため以下の処理を行う．

(9) N の要素平均に対して比較的小さい数をランダムに生成し，そのサイズの空タスクをコネクションに挿入することにより，コネクションを短時間 sleep させる．N_s がある場合，(3) へ

5. 評価及び考察

提案する複製選択，複製作成アルゴリズムに対して一様ランダムにファイルサイズ，ファイルのノードに対する配置を決定するシュミレータを作成した．評価に関するパラメータとして，ファイルの数，ノードの数，コネクションの数，複製の存在しやすさ，ノードに対するファイルの偏りやすさ，ファイルの最大値を設定した．

5.1 評価プログラム

作成したプログラムは与えられたパラメータに従い，一様ランダムにファイルサイズ，ファイルの配置を決定するものである．以下のパラメータをとる．

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．使用可能バンド幅の分割数．

Max File Size ファイルサイズの最大値

Replication ファイル複製の存在しやすさを表す．0.99〜0.01 の幅で設定する．

Node Bias ファイルのノードに対する偏りやすさを表す．0.99〜0.01 の幅で設定する．

Replication は，ファイル複製が存在する確率として実現し，ファイル f に対して p 個の複製が $(Replication)^{p-1}$ の確率で存在することとした．

Node Bias は，ノード列 $_1 \in \{1, \cdots, m\}$ に対して，先頭から q 番目のノードにファイル f が割り当てられる確率が $(Node\ Bias)^q$ であるとした．

5.2 評価内容

ファイル数，ノード数，コネクション数，Replication，Node Bias が，スケジューリングアルゴリズムにより計算されたシステム全体の予想転送終了時間に与える影響を調べることにより，提案アルゴリズムを評価する．提案アルゴリズムの比較対象として以下のものについて評価する．

Simple List ノードに対する複製の割り当て状況や，ファイルの配置状況を考慮しないリストアルゴリズム．

Select ファイルの配置を考慮し，複製の選択を行う．しかし，動的な複製選択は行わない．

Create 複製選択とともに，動的な複製作成を行う．

Ideal 総ファイルサイズをコネクション数で割った値．コネクションを全て，休まずに使えている状態を表す．

評価方法としてはパラメータの基本値として，

- $Number\ of\ Files = 1000$
- $Number\ of\ Nodes = 100$
- $Number\ of\ Connections = 50$
- $Max\ File\ Size = 5000$

を設定し，ファイル数，ノード数，コネクション数を変化させた．

5.3 評価結果

パラメータを変化させ得た結果より，特に転送予想時間変化と相関がみられたファイル数を変化させた場合，コネクション数を変化させた場合について考察する．

ここで，ノード数を変化させた場合について簡単に述べる．今回のシミュレーション結果からは予想転送時間とノード数との間に大きな相関は見られなかったが，コネクション変化の章で詳しく述べる理由によりノード数が比較的小さい時（100 - 200 程度）のときは，SimpleList と Select においてノード数と予想転送時間との間に反比例の関係がみられた．

5.3.1 ファイル数を変化させた場合

ファイル数を変化させた場合の，予想転送時間を**図 3(a) 図 3(b) 図 3(c)** に示す．図中の"repl"は複製が存在しやすい場合を意味し，"nrepl"は複製が存在しにくい場合を意味する．

複製選択を行った場合，図 3(a) のとき，SimpleList と比較して約 25% ，図 3(b) のとき，SimpleList と比較して約 57% の時間とすることができた．

また，Create アルゴリズムは，複製が存在しない場合，図 3(a) で最大で SimpleList の 25% の時間で転送を終了することができるなど，かなり転送時間を短くすることができている．

今回，複製が存在する場合，Create は Select よりも性能が低下している．このとき，4.4 で述べたように隠蔽不可能なオーバーヘッドが生じるが，今回の最小限の実装では，この場合の処理の最適化までは行っていないためこのような結果になったと考えられる．このような場合の最適化は今後の課題とする．

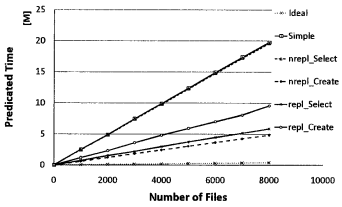
5.3.2 コネクション数を変化させた場合

複製が存在しない場合，**図 4(a)**，**図 4(c)** から，Create アルゴリズムにより効果的に複製作成を行うことができ，Ideal に十分に近い値を得ることができている．また，ファイル複製が十分に存在する場合，5.3.1 節で述べた理由により，今回のパラーメータでは Create は Select よりも予想転送時間を短くできていないという結果になった．しかしながら，コネクション数を増やした時に SimpleList，Select がある値で収束するのに対し，Create は転送時間を短くすることができている．(図 4(a)，**図 4(b)**)

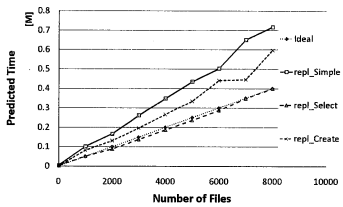
6. 関連研究

本稿では，ファイル転送タスクをヒューリスティックを用いてスケジューリングを行うアルゴリズムについて述べた．今回扱った独立なタスクのスケジューリングについては盛んに研究がおこなわれており，cpu プロセススケジューリングや広域環境でのタスクスケジューリングなどに応用されている．

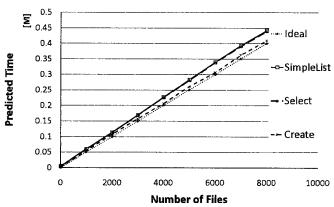
Tracy ら[5] は，ヘテロな分散環境において独立タスクスケジューリングの有用性を評価するために，独立タスクを扱う 11 種のアルゴリズムについてシミュレー



(a) ノードに偏りがある場合

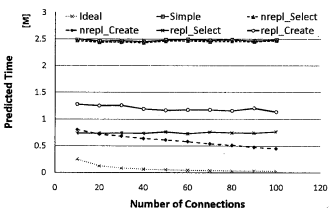


(b) ノードに偏りがなく，複製がある場合

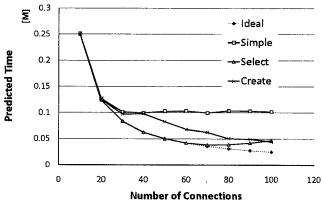


(c) ノードに偏りがなく，複製がない場合

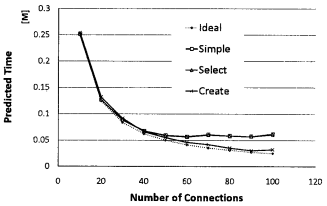
図 3 ノード数=100，コネクション数=50 の場合におけるファイル数の変化



(a) ノードに偏りがある場合



(b) ノードに偏りがなく，複製がある場合



(c) ノードに偏りがなく，複製がない場合

図 4 ファイル数=1000，コネクション数=50 の場合におけるコネクション数の変化

ションを行い評価を行っている．

本稿では，独立なタスクや分割可能なタスクを扱ったが，須田[4] は，これらのタスクスケジューリングを含むヘテロな並列環境において有用なスケジューリング手法について非常に詳しく述べている．

7. おわりに

本稿では我々が提案するクラスタ間並列ファイル転送システムにおけるファイル転送タスクのスケジューリングに関して述べた．クラスタ内の複数ノードに分散するファイルを効率的に転送するために必要となる問題について分析し，リストアルゴリズムを応用するスケジューリングアルゴリズムを実装し評価を行った．ノードに対しファイルサイズ分布の偏りがある場合には，複製作成タスクを外部転送の裏で行うようスケジューリングし，十分に予想される転送時間を短縮することができた．

今後の課題として，複製が十分にありはじめから平均化されている場合には十分な性能が出ていない．このためアルゴリズム内のパラメータの最適化や，効果的なヒューリスティックの利用，タスク量の計算の精度向上などが必要であると考える．

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 19024009 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) Allcock, B. Bester, J. Bresnahan, J. Chervenak, A.L. Kesselman, C. Meder, S. Nefedova, V. Quesnel, D. Tuecke, S. Foster, I.: Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing, Eighteenth IEEE Symposium on Mass Storage Systems and Technologies, April 2001, pp13-13

2) Madduri, R.K. Hood, C.S. Allcock, W.E.: Reliable file transfer in Grid environments, Proceedings of 27th Annual IEEE Conference on Local Computer Networks, NOV2002, pp737-738

3) Stewart, G.A., Cameron, D. Cowan, G.A. Cance, G.Mc.: Storage and data management in EGEE, Proceedings of the fifth Australasian symposium on Grid Computing and e-Research (AusGrid 2007),pp69-77

4) 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，Vol47, NoSIG_18(ACS_16)(20061115),pp. 92-114.

5) T. D. Braun , H. J, Siegel , N. Beck , L. L. Boloni , M. Maheswaran , A. I. Reuther , J. P. Robertson , M. D. Theys , B. Yao , D. Hensgen , R. F. Freund, A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, Journal of Parallel and Distributed Computing, v.61 n.6, p.810-837, June 2001