セキュアプロセッシングにおけるファイル分散配置による通信負荷改善の効果に関する研究

平田 智紀[1]　稲元 勉[2]　樋上 喜信[2]　小林 真也[2]

概要：本稿は，グリッドコンピューティングにおける信頼性の向上を目指す手法である処理の多重化の問題点である，処理の依頼を行う計算機に集中する通信負荷の軽減を目的とする．処理の多重化は，同一の処理内容を複数の計算機に依頼し，その処理結果を比較することで，処理結果が正しいか否かを判別する手法である．この手法では，プログラムや入力データを配布するためにグリッド上の計算機と繰り返し通信する必要から，処理を依頼する計算機には多大な通信負荷がかかる．ここで，プログラムや入力データを秘密分散法によってシェアへと変換し，処理を依頼するものとは異なる計算機へシェアを一時的に保管し，それらから配布することを考える．このようにすることで，シェア保管用の計算機に対する秘匿性を保ちつつ，処理を依頼する計算機の通信負荷を軽減できる．この際の，各計算機にかかる通信負荷を調査し，通信負荷が改善されたことを示す．

A Study on Improvement on Communication Load in Secure Processing Brought by Distributing Files

Tomonori Hirata[1]　Tsutomu Inamoto[2]　Yoshinobu Higami[2]　Shin-ya Kobayashi[2]

1. はじめに

インターネットと高性能なパーソナルコンピュータの普及により，無数のコンピュータが接続された世界的規模のネットワークコンピューティング環境が出現している．この環境を活用することで，ネットワーク上に分散している計算機を統合し，透過的なサービスを利用可能にする手法であるグリッドコンピューティング [1] は，高い処理性能を得ることができると期待される．

グリッドコンピューティングは，企業や研究室といったイントラネット上にある，管理され信頼できる計算機を利用するインターナルグリッドと，インターネット上の管理されていない計算機も利用するエクスターナルグリッドに分けられる．インターナルグリッドは，利用可能な計算機が少ないが，それらはグリッドの利用者によって管理されているものであり，グリッドの処理結果を信頼することができる．一方，エクスターナルグリッドは，利用可能な計算機が膨大であるため，潜在的な処理能力は高い．しかし，グリッドの利用者と計算機の所有者が異なり，グリッドを構成する計算機を信用できるとは限らないため，処理内容／処理データに関する秘匿性や，処理結果の信頼性は低い．このようなエクスターナルグリッドの欠点への対策として，著者らは，処理内容の隠蔽 [2][3] や処理結果の信頼性向上 [4]，意図的に誤った処理結果を返す計算機の検出 [4] などを目指して，“セキュアプロセッシング” という取り組みを進めている．

計算機から返される処理結果が正しいか否かを判別するために，同一の処理を複数の計算機で処理し，それらの処理結果を比較する手法がある．この手法を “処理の多重化” と呼ぶ．処理の多重化を行うことで，誤った処理結果を受け入れる可能性が低くなり，信頼性の高い処理結果を得ることができる．その一方で，処理を依頼する計算機の通信負荷が重くなることが考えられる．

[1] 愛媛大学工学部
Faculty of Engineering, Ehime University, Bunkyo-cho 3, Matsuyama, Ehime, 790–8577, Japan
（*2014 年 4 月より株式会社 STNet）
[2] 愛媛大学大学院理工学研究科
Graduate School of Science and Engineering, Ehime University, Bunkyo-cho 3, Matsuyama, Ehime, 790–8577, Japan

本稿では，処理を依頼する計算機の通信負荷を軽減する手法を述べ，その手法の有効性を示す．

2. 処理の多重化と秘密分散法

2.1 処理の多重化

エクスターナルグリッドは，処理を行うノードが信頼できるとは限らないため，依頼された処理に対して誤った処理結果を返すことがある．その原因として，処理を行うノードのハードウェアの故障やソフトウェアのバグといった意図されないものや，悪意をもってプログラムや入力データを変更した意図的なものが考えうる．このような，誤った処理結果を返すことを “改ざん” と呼ぶ．改ざんに対する対策の一つとして，同一の処理を複数の計算機で行い，処理結果を比較することでその正しさを判別するという “処理の多重化” がある．このとき，エクスターナルグリッドにおいて，処理を依頼しプログラムや入力データを配布する計算機を “管理ノード”，計算資源を提供し依頼された処理を行う計算機を “処理ノード”，同じ処理を依頼する計算機の数を “多重度” と呼ぶ [2]．一般に，多重度を高くするほど，誤った処理結果が導出される可能性が低くなり，信頼性の高い処理結果を得ることができる．一方で，多重度が高いほど，管理ノードは多くの処理ノードに同じ処理を依頼することになる．同一の処理を行うすべての処理ノードから処理結果を受け取ることは，管理ノードへの通信負荷の集中をもたらす．その対策として，処理結果を利用するつぎの処理ノードが，処理結果の受け取りと比較を行う方法がある．この方法は，処理結果の受け取り・比較を行う計算機が，受け取った結果に対してさらに何らかの処理を施す場合，妥当である．本稿では，この方法を取り入れた場合を対象として，評価・考察を行う．

多重度 5 である処理の多重化における，ファイル配布時の通信の例を図 1 に示す．また，処理結果の受け取りと比較を行いながら，処理が実行されていく際の通信の様子を図 2 に示す．

単純な処理の多重化では，管理ノードは文献 [3] で示された処理目的の隠蔽法をプログラムに施し，それによって得られたプログラム断片や入力データ（以後，ファイル）をすべての処理ノードへ配布する．ここで，信頼性を高く保つために多重度を高くすると，処理ノードの数が多くなることから，ファイル配布にともなう通信負荷が管理ノードに集中する．これが原因で，グリッドシステム全体としてのスループットが低くなるという問題が生じる．この問題の解決策として，管理ノードの通信負荷を他の計算機に分担させることが考えられる．

2.2 秘密分散法

秘密分散法は，秘密情報を分散管理するための方法であり，分散した情報を集めても，その数が一定数以下である場合には元の秘密情報を復元できないという点をおもな特徴とする．秘密分散法を用いることで，元のプログラムを秘密にしたまま，ファイルを分散することができる [5][6]．

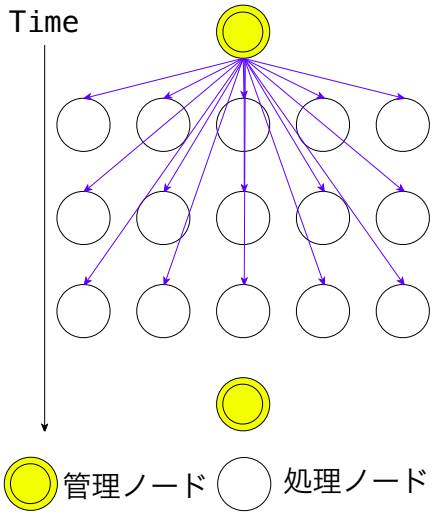

図 1 処理の多重化（プログラム断片の配布）

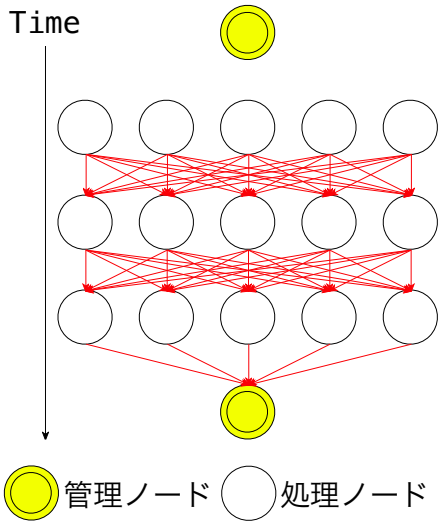

図 2 処理の多重化（出力結果の受け渡し）

秘密分散法の特徴を利用することで，処理の多重化が有する問題の一つである，管理ノードにかかる重い通信負荷を，他の計算機に分担させることが可能になる．まず，一時的にファイルを保管しておく計算機を “保管ノード” と呼び，保管ノードを複数用意することを考える．このとき，保管ノードを管理者が用意する場合と，保管ノードとしてエクスターナルグリッド上の他者が管理する計算機を利用する場合が考えられる．前者は，複数台の計算機を用意する必要があり，コストがかかる．その一方，後者は，コストがかからないが，インターネット上の不特定の計算機に保管するため，ファイルの情報を秘密にしておく必要がある．ここで，そのような計算機にファイルをそのまま保管しておくのではなく，秘密分散法によって作られた分散情報を保管する．これにより，そのような計算機に対して秘

匿性を保ったまま，管理ノードへの通信負荷の集中を回避できる．

秘密分散法には，(k,n) しきい値秘密分散法と (k,L,n) しきい値ランプ型秘密分散法がある．それぞれの方法について，以下に説明する．

(k,n) しきい値秘密分散法:

秘密にしたい情報を秘密情報 S とする．秘密情報 S を n 個のシェアと呼ばれる分散情報 $W_i\,(1 \leq i \leq n)$ に分散符号化する．これら n 個のシェアのうち，任意の k 個から秘密情報を復元できる．ただし，どのシェアを $k-1$ 個集めても秘密情報 S を復元できない．また，シェアの一つ一つを見ても秘密情報の部分的な情報が分からないようになっている．

(k,n) しきい値秘密分散法では，シェア W_i はつぎの $k-1$ 次多項式 (1) によって算出される．

$$W_i = S + a_1 x_i^1 + a_2 x_i^2 + \cdots + a_{i-1} x_i^{i-1} \pmod{p}. \quad (1)$$

ここで，p は S, n よりも大きな素数であり，$a_1, \ldots, a_{i-1}$ は p よりも小さな乱数値である．式 (1) における x_i を W_i のシェア ID という．復元の際には，x_i, W_i から S が求められる．

(k,n) しきい値秘密分散法では，シェアの合計サイズが秘密分散する元の秘密情報の n 倍程度になる．よって，この手法で作られた分散情報を保管ノードへ保管した場合，保管ノード・処理ノードに多大な通信負荷がかかる．

(k,L,n) しきい値ランプ型秘密分散法:

L を任意の整数とし，秘密情報 S を L 個に分割したものを秘密情報 $S^{(L)} := \langle s_0, s_1, \ldots, s_{L-1} \rangle$ とする．$S^{(L)}$ を，(k,n) しきい値秘密分散法と同じように，n 個のシェアへ分散符号化する．任意の k 個以上のシェアから秘密情報 $S^{(L)}$ を復元できるが，任意の $k-L$ 個以下のシェアからは $S^{(L)}$ に関する情報をまったく得ることができない．また，シェアの数が $k-L-1$ 個以上 $k-1$ 個以下の場合には，シェアの増加にともなって，$S^{(L)}$ に関して得られる情報が増えていく．

(k,L,n) しきい値ランプ型秘密分散法では，p より小さな乱数値 $a_L, a_{L+1}, \ldots, a_{i-1}$ を用いて，つぎの $k-1$ 次多項式 (2) によりシェア W_i が算出される．

$$W_i = s_0 + s_1 x_i^1 + s_2 x_i^2 + \cdots + s_{L-1} x_i^{L-1} + a_L x_i^L + \cdots + a_{i-1} x_i^{i-1} \pmod{p}. \quad (2)$$

式 (2) における x_i を W_i のシェア ID という．復元の際には，x_i, W_i から S が求められる．

(k,L,n) しきい値ランプ型秘密分散法では，シェアの合計サイズが秘密分散する元の秘密情報の n/L 倍程度になる．よって，(k,L,n) しきい値ランプ型秘密分散法は，(k,n) しきい値秘密分散法に比べて，秘匿に対する安全性は劣るものの，通信負荷を $1/L$ 程度に軽減できると考えられる．

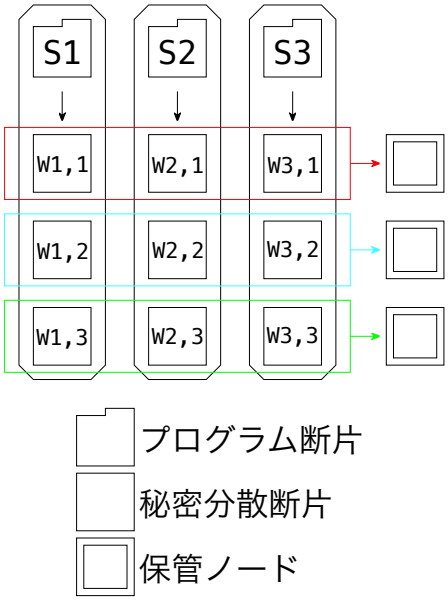

図 3　秘密分散法を用いたプログラム断片の保管ノードへの保存

3. 通信負荷の軽減手法

通信負荷の軽減手法は，(k,L,n) しきい値ランプ型秘密分散法を利用してファイルを分散配置し，管理ノードの通信負荷を軽減するというものである．本稿では，プログラムに処理内容の隠蔽を施したものをプログラム断片と呼び，プログラム断片を秘密分散法により分散化したものを秘密分散断片と呼ぶ．同様に，入力データを秘密分散法により分散化したものを秘密分散データと呼ぶ．

以下に，通信負荷の軽減手法の手順を示す．

(1) 秘密分散断片の生成

管理ノードは，処理目的の隠蔽法 [3] を施し，x 個のプログラム断片 $P_i\,(i = 1, 2, \ldots, x)$ を作成する．そして，(k,L,n) しきい値ランプ型秘密分散法によりプログラム断片 P_i をそれぞれ n 個の秘密分散断片 $W_{i,j}\,(j = 1, 2, \ldots, n)$ に変換する．

(2) 秘密分散断片の配布

管理ノードは保管ノードとなる n 個のノードを選ぶ．秘密分散断片 $W_{i,j}$ を保管ノード H_j に配布する．秘密分散法を利用して保管ノードにプログラム断片を保存する一例を図 3 に示す．

(3) 保管ノード表の作成・配布

プログラム断片 P_i に対する保管ノード表 L_i を生成する．ここで，保管ノード表 L_i には，プログラム断片 P_i を復元するために必要な秘密分散断片のハッシュ値・シェア ID と，その秘密分散断片を保管している

保管ノードを識別するノード ID が書かれている．管理ノードは，i 段目のすべての処理ノードに保管ノード表 L_i を配布する．

(4) プログラム断片の復元

処理ノードは，保管ノード表に書かれている保管ノードに，自身が必要とする秘密分散断片を要求する．そして，取得した秘密分散断片のハッシュ値と保管ノード表に書かれているハッシュ値が同値であるか確認する．同値であれば高い確率で正しい秘密分散断片であり，違う値ならば不正な秘密分散断片である．n 個のうち k 個以上の正しい秘密分散断片を取得すれば，実行するプログラム断片を復元できるようになる．

入力データも，プログラム断片と同様に，(k, L, n) しきい値ランプ型秘密分散法を用いて秘密分散データへ変換したあと，保管ノードへ配布する．

以下に，通信量・耐解析・耐改ざんの 3 点に関する，通信負荷の軽減手法の特徴をまとめる．

通信量:

(k, L, n) しきい値ランプ型秘密分散法を用いるため，秘密分散断片や秘密分散データの合計サイズが元のプログラム断片や入力データの k/L 倍になる．この比率は，(k, n) しきい値秘密分散法に対して $1/L$ である．秘密分散断片のサイズの総量と秘匿性を比較したものを表 1 に示す．

耐解析:

保管ノードに保管されている秘密分散断片・秘密分散データは秘密分散法によって生成されたものであるため，保管ノード単体では，元のプログラム断片や入力データの情報を得ることはできない．また，本来，秘密分散断片や秘密分散データを必要としないノードには，保管ノード表が配布されないため，秘密分散断片や秘密分散データを所持している保管ノードを探し当てることは容易ではない．このことから，処理を依頼された処理ノード以外が元のプログラム断片や入力データを復元することはきわめて難しい．

一方で，しきい値 k 以上の保管ノードが共謀した場合，元のプログラム断片を復元できるため，しきい値 k を慎重に決める必要がある．さらに，(k, L, n) しきい値ランプ型秘密分散法では，所持する秘密分散断片の数が $k-L-1$ 個以上 $k-1$ 個以下の場合，その増加にともなって得られる情報が増加するという特徴を持つ．よって，(k, n) しきい値秘密分散法に比べて，秘匿性は低くなる．

耐改ざん:

処理ノードが保管ノードに秘密分散断片や秘密分散データを要求した際に，改ざんされたものが配布されたとする．この場合，秘密分散断片や秘密分散データのハッシュ値と保管ノード表に書かれたハッシュ値が同値であるかを確認し，一致しなかったものに関しては，改ざんされたものと見なすことができる．

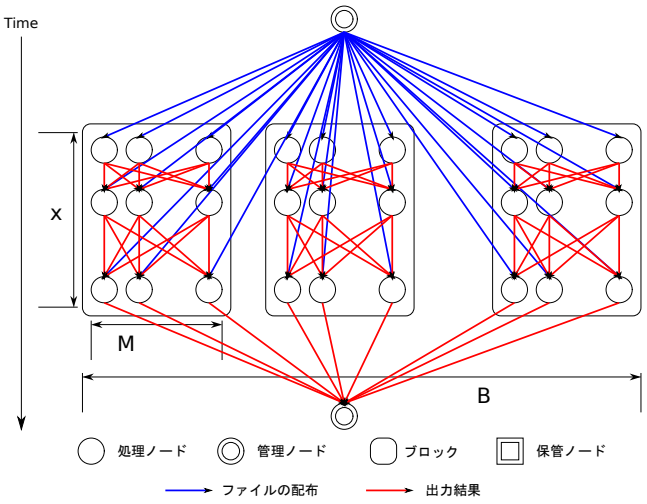

図 4　従来の保管ノードを用いないグリッド構成図

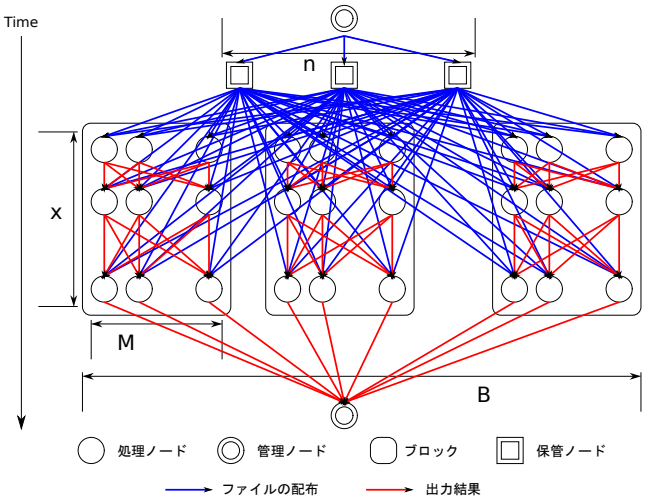

図 5　管理ノードの通信負荷の軽減手法を用いたグリッド構成図

4. 通信負荷の算出

4.1 前提

エクスターナルグリッド上の計算機は膨大であり，処理ノードや保管ノードとして利用することができる．また，入力データをかえて同一の処理を複数回行うことを想定し，プログラム断片（秘密分散断片）のサイズ，各処理ノードに対する入力データ（秘密分散データ）のサイズはすべて等しいものとする．

4.2 記号の定義

従来の保管ノードを用いない場合，通信負荷の軽減手法を用いた場合のセキュアプロセッシングにおけるグリッド構成図を，それぞれ図 4, 5 に示す．

以下の定量的解析で用いられる変数・記号を定義する．

- D: プログラム断片のサイズ [KB]
- I: 入力データのサイズ [KB]

表 1　秘密分散断片の比較

項目	(k,n) しきい値秘密分散法による秘密分散断片	(k,L,n) しきい値ランプ型秘密分散法による秘密分散断片
秘匿性	高い	(k,n) しきい値秘密分散法に比べて低い
秘密分散断片のサイズの総量	プログラム断片のサイズの総量の k 倍	プログラム断片のサイズの総量の k/L 倍

- x: プログラムの段数[*1]
- M: 多重度
- n: 保管ノードの数
- k: 復元に必要な秘密分散断片の数
- B: ブロック数（並列処理する入力データ数 [2]）
- H: 保管ノード表とハッシュ値表の合計サイズ [KB]
- L: (k,L,n) しきい値ランプ型秘密分散法における秘密情報の分割数

4.3　通信負荷の算出

従来の保管ノードを用いない場合，通信負荷の軽減手法用いた場合において，管理ノード・保管ノード・処理ノードのそれぞれにかかる通信負荷を比較するため，それらを定式化する．

管理ノードは，ファイルの配布，および処理結果の受け取りを行う．これらのうち，後者の通信量は，従来の保管ノードを用いない場合と通信負荷の軽減手法を用いた場合で同一である．よって，前者の，ファイルの配布にかかる通信負荷のみを調べる．

保管ノードは，管理ノードからのファイルの受け取り，および処理ノードへのファイルの配布を行う．これら二つのうち，前者にかかわる通信量はきわめて小さいため，ファイルの配布にかかる通信負荷のみを調べる．

処理ノードは，ファイルの受け取りと次段への入力データの配布を行うため，それぞれの通信にかかわる通信負荷を調べる．

従来の保管ノードを用いない場合では，図 4 のように，管理ノードはすべての処理ノードへプログラム断片を配布する．全処理ノード数は $B \times M \times x$ であるため，プログラム断片を配布する際の管理ノードにかかる通信量は $D \times B \times M \times x$ と表される．また，管理ノードは，処理に必要な入力データを，各ブロックの 1 段目の処理ノードへ配布する．1 段目の処理ノード数は $B \times M$ であるため，入力データを配布する際に管理ノードにかかる通信量は $I \times B \times M$ と表される．

また，1 段目の処理ノードは，プログラム断片と入力データを管理ノードから受け取り，次段の入力データ配布を行う．2 段目以降の処理ノードは，プログラム断片と入力データ（前段の出力結果）を受け取り，次段の入力データ配布を行う．

したがって，従来の保管ノードを用いない場合の各ノードにかかる通信負荷は以下のようになる．

- 管理ノードにかかる通信負荷

$$D \cdot B \cdot M \cdot x + I \cdot B \cdot M. \tag{3}$$

- 1 段目の各処理ノードにかかる通信負荷

$$D + I + M \cdot I. \tag{4}$$

- 2 段目以降の各処理ノードにかかる通信負荷

$$D + 2 \cdot M \cdot I. \tag{5}$$

(k,n) しきい値秘密分散法による通信負荷の軽減手法を用いた場合，管理ノードは，図 5 のように，保管ノードへ秘密分散断片および秘密分散データを，処理ノードへ保管ノード表を，それぞれ配布する．秘密分散断片のサイズはプログラム断片のサイズと等しいため，一つの保管ノードが配布する秘密分散断片のサイズは $D \times x$ と表される．よって，管理ノードが秘密分散断片を保管ノードへ配布する際の通信負荷は，$D \times n \times x$ と表される．秘密分散データのサイズは $I \times B$ であるため，秘密分散データを配布する際に管理ノードにかかる通信負荷は $I \times B \times n$ と表される．さらに，すべての処理ノードへ保管ノード表を配布する際の通信負荷は $H \times B \times M \times x$ と表される．

保管ノードは処理ノードの要求によって，秘密分散断片と秘密分散データを配布する．プログラム断片・入力データと秘密分散断片・秘密分散データのサイズは等しく，保管ノード n 個のうち k 個が配布を行うため，保管ノードが秘密分散断片を配布する際の通信負荷は $D \times B \times M \times x \times (k/n)$ と表される．同様に，保管ノードが秘密分散データを配布する際の通信負荷は $I \times B \times M \times (k/n)$ と表される．

また，1 段目の処理ノードは，k 個の保管ノードからプログラム断片と入力データ，管理ノードから保管ノード表をそれぞれ受け取る．2 段目以降の処理ノードは，k 個の保管ノードからプログラム断片，前段から入力データ（前段の出力結果），管理ノードから保管ノード表を受け取り，次段へ入力データの配布を行う．

したがって，(k,n) しきい値秘密分散法による通信負荷の軽減手法を用いた場合の各ノードにかかる通信負荷は以下のようになる．

- 管理ノードにかかる通信負荷

[*1] 生成した x 個のプログラム断片 $P_i\,(i = 1, 2, \ldots, x)$ は処理の依存関係によって実行順序が決まっており，$P_1 \to P_2 \to \cdots \to P_x$ の順番で実行する必要があるとする．このときの x の値 [2].

$$D \cdot n \cdot x + I \cdot B \cdot n + H \cdot B \cdot M \cdot x. \quad (6)$$

- 各保管ノードにかかる通信負荷

$$D \cdot B \cdot M \cdot x \cdot (k/n) + I \cdot B \cdot M \cdot (k/n). \quad (7)$$

- 1 段目の各処理ノードにかかる通信負荷

$$k \cdot D + k \cdot I + H. \quad (8)$$

- 2 段目以降の各処理ノードにかかる通信負荷

$$k \cdot D + 2 \cdot M \cdot I + H. \quad (9)$$

(k, L, n) しきい値ランプ型秘密分散法による通信負荷の軽減手法では，(k, n) しきい値秘密分散法による通信負荷の軽減手法と同様に通信を行う．ただし，秘密分散断片，秘密分散データのサイズがそれぞれ $1/L$ になる．

したがって，(k, L, n) しきい値ランプ型秘密分散法による通信負荷の軽減手法を用いた場合の各ノードにかかる通信負荷は以下のようになる．

- 管理ノードにかかる通信負荷

$$(D/L) \cdot n \cdot x + (I/L) \cdot B \cdot n + H \cdot B \cdot M \cdot x. \quad (10)$$

- 保管ノードにかかる通信負荷

$$(D/L) \cdot B \cdot M \cdot x \cdot (k/n) + (I/L) \cdot B \cdot M \cdot (k/n). \quad (11)$$

- 1 段目の各処理ノードにかかる通信負荷

$$k \cdot (D/L) + k \cdot (I/L) + H. \quad (12)$$

- 2 段目以降の各処理ノードにかかる通信負荷

$$k \cdot (D/L) + 2 \cdot M \cdot (I/L) + H. \quad (13)$$

各項目に対して，従来の保管ノードを用いないセキュアプロセッシングと通信負荷の軽減手法を用いたセキュアプロセッシングにおいて，通信量の点で比較する．また，各項目の式 (3)–(13) における解析結果を比較・考察する．

5. 解析結果

保管ノードを用いず処理の多重化を行う手法，(k, n) しきい値秘密分散法による通信負荷の軽減手法，(k, L, n) しきい値ランプ型秘密分散法による通信負荷の軽減手法を，それぞれ既存手法 1，既存手法 2，提案手法と呼び，各手法を用いた場合の通信負荷を調べる．通信負荷は，管理ノード，保管ノード，1 段目の処理ノード，2 段目以降の処理ノードという 4 種類のノードごとに調べる．また，各種ノードについて，ブロック数 (B)，プログラム断片のサイズ (D)，保管ノード表のサイズ (H)，入力データのサイズ (I)，多重度 (M)，保管ノード数 (n)，プログラムの段数 (x) という 7 種類のパラメータを変えることが通信負荷へ与える影響を，それぞれ調べる．変更しないパラメータの値は，表 2 に示すよう設定する．

表 2　パラメータの既定値

項目	設定値
プログラム断片のサイズ [KB]:D	300
入力データのサイズ [KB]:I	100
段数:x	10
多重度:M	3
保管ノードの数:n	20
復元に必要な秘密分散断片の数:k	11
ブロック数:B	300
保管ノード表とハッシュ値表の合計サイズ [KB]:H	4
(k, L, n) しきい値秘密分散法における L:L	5

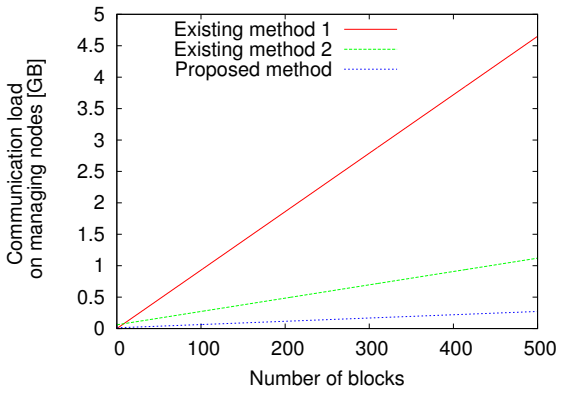

図 6　ブロック数に対する管理ノードにかかる通信負荷の比較

5.1 管理ノードにかかる通信負荷

図 6, 7, 10, 12 より，ブロック数，プログラム断片のサイズ，多重度，段数の増加にともない，既存手法 1 では通信負荷が増えていることから，提案手法による効果を見込めることがわかる．

一方で，図 8, 9, 11 より，保管ノードの数，入力データのサイズ，保管ノード表のサイズについては，その増加にともない通信負荷が増えていることから，提案手法の効果を見込めない．

提案手法による通信負荷の軽減効果が見込めないパラメータについて，既存手法 1 との比較を行うと，$M > n/L$，つまり L を n/M より大きくすることで，式 (3) における入力データサイズの係数が式 (10) よりも大きくなるため，入力データのサイズが大きくなるほど提案手法の効果を見込める．

5.2 保管ノードにかかる通信負荷

保管ノードについては，図 13–19 より，保管ノードを用いない既存手法 1 を除いて，すべての項目において提案手法の効果を見込める．

5.3 処理ノードにかかる通信負荷

処理ノードについては，図 20–33 より，保管ノードを用いない既存手法 1 と比べて，すべての項目において通信負荷が増えており，提案手法の効果を見込めない．しかし，

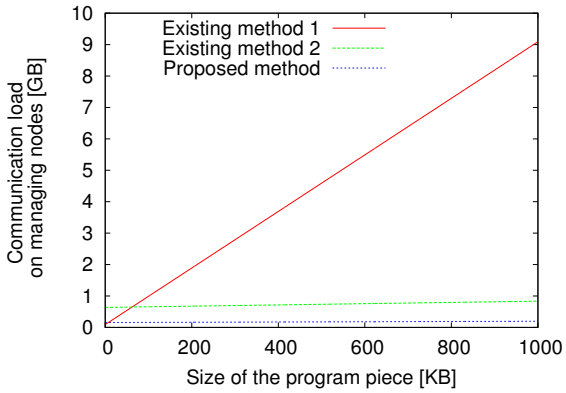

図 7　プログラム断片のサイズに対する管理ノードにかかる通信負荷の比較

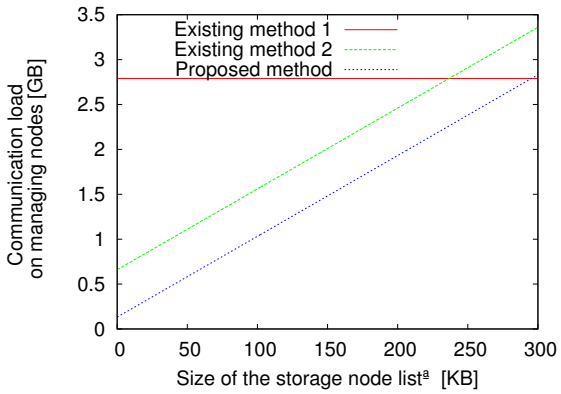

図 8　保管ノード表のサイズに対する管理ノードにかかる通信負荷の比較

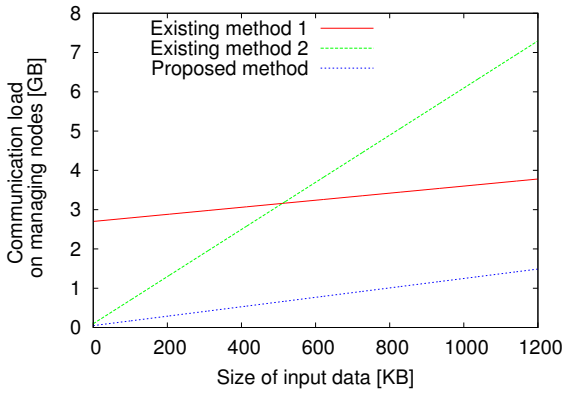

図 9　入力データのサイズに対する管理ノードにかかる通信負荷の比較

既存手法 2 に比して通信負荷を $1/L$ とできることから，提案手法の効果を見込める．

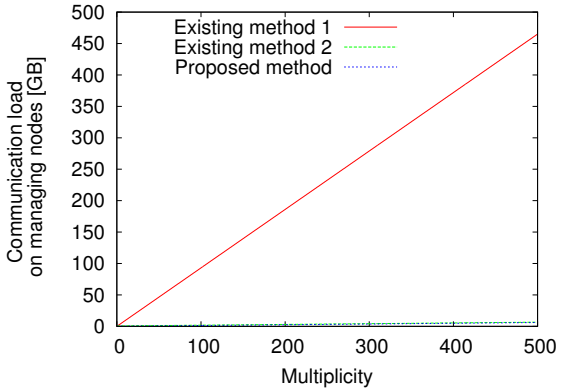

図 10　多重度に対する管理ノードにかかる通信負荷の比較

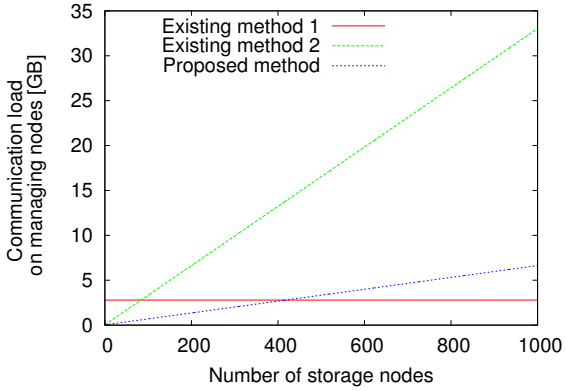

図 11　保管ノードの数に対する管理ノードにかかる通信負荷の比較

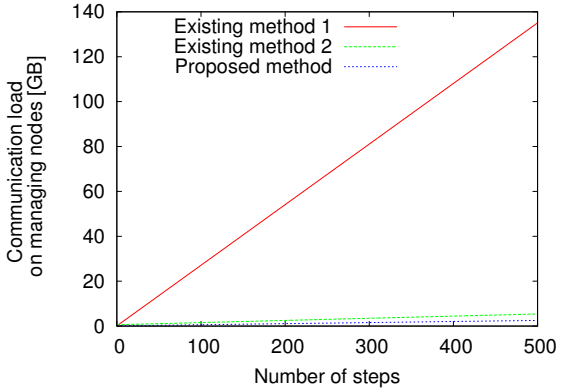

図 12　段数に対する管理ノードにかかる通信負荷の比較

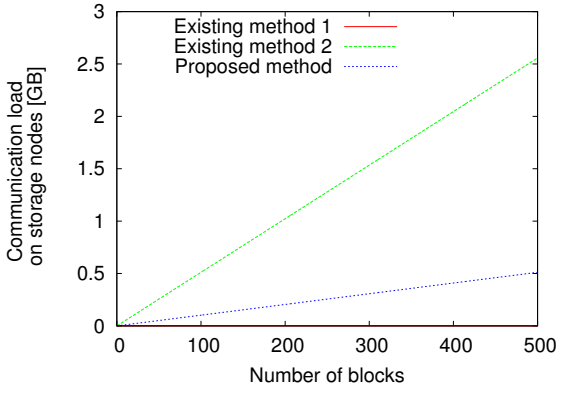

図 13　ブロック数に対する保管ノードにかかる通信負荷の比較

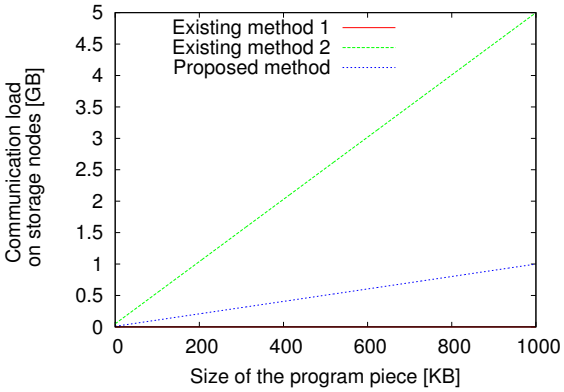

図 14　プログラム断片のサイズに対する保管ノードにかかる通信負荷の比較

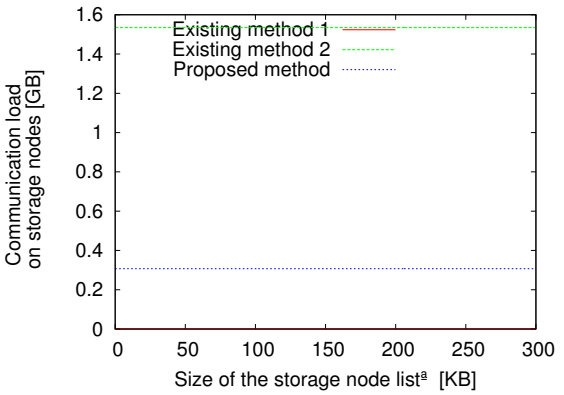

図 15　保管ノード表のサイズに対する保管ノードにかかる通信負荷の比較

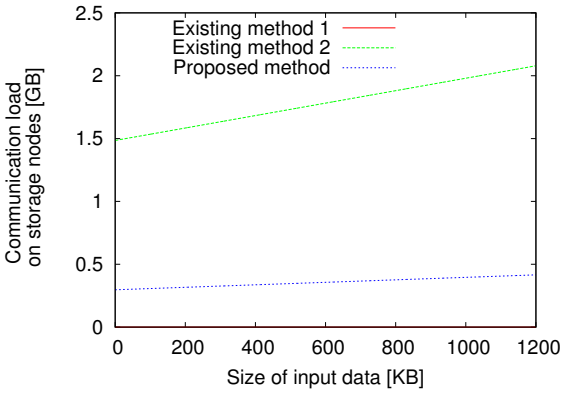

図 16　入力データのサイズに対する保管ノードにかかる通信負荷の比較

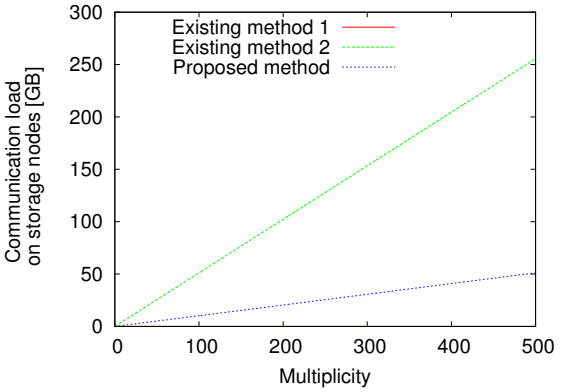

図 17　多重度に対する保管ノードにかかる通信負荷の比較

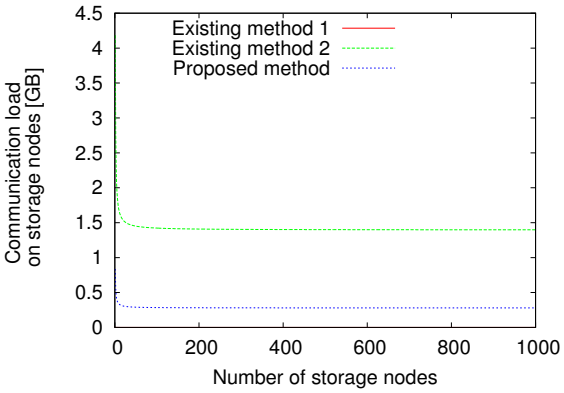

図 18　保管ノードの数に対する保管ノードにかかる通信負荷の比較

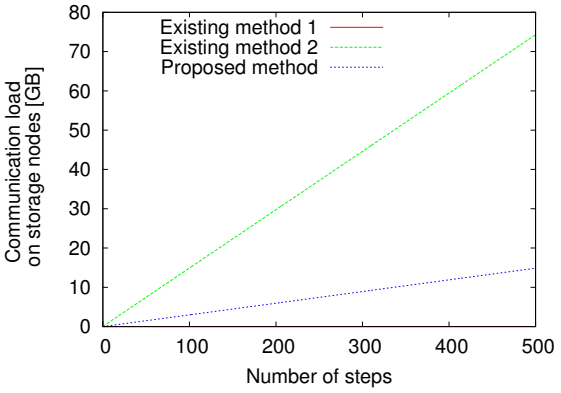

図 19　段数に対する保管ノードにかかる通信負荷の比較

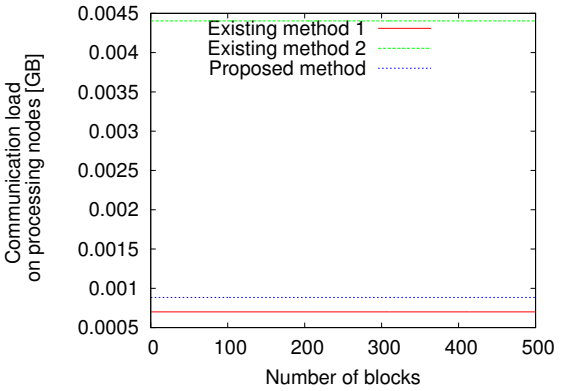

図 20　ブロック数に対する一段目の処理ノードにかかる通信負荷の比較

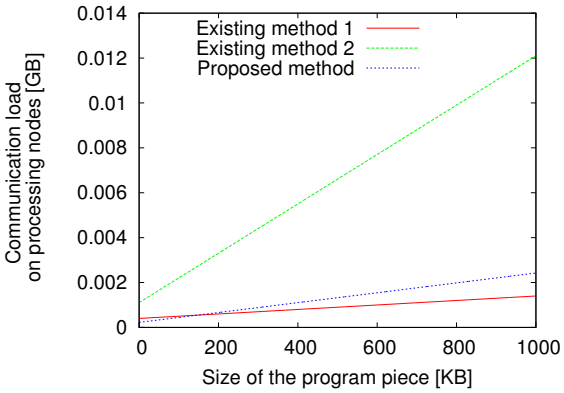

図 21　プログラム断片のサイズに対する一段目の処理ノードにかかる通信負荷の比較

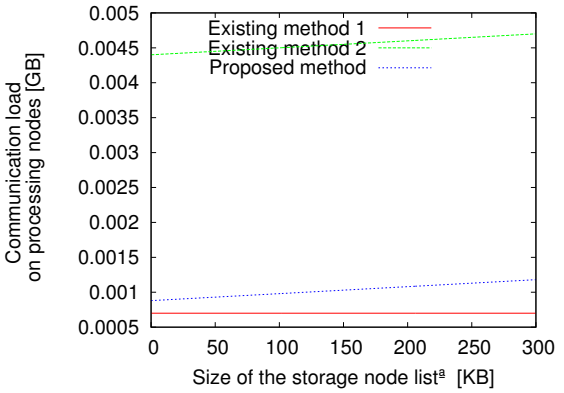

図 22　保管ノード表のサイズに対する一段目の処理ノードにかかる通信負荷の比較

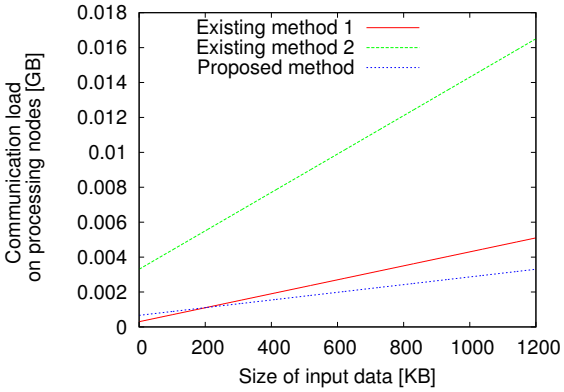

図 23　入力データのサイズに対する一段目の処理ノードにかかる通信負荷の比較

5.4　多重度

以上の結果から，多重度を高くすることが，管理ノード・処理ノードの通信負荷を増すことがわかった．ここで，現実的には，多重度をどの程度にすればよいかを考察する．

ブロック内のプログラム断片は依存関係により実行順序が決まっている．グリッドの使用を想定するようなプログラムにおいては，段数が多くなり，かつ，全体として正しくない結果を一度も採用しない確率は，正しい結果を採用

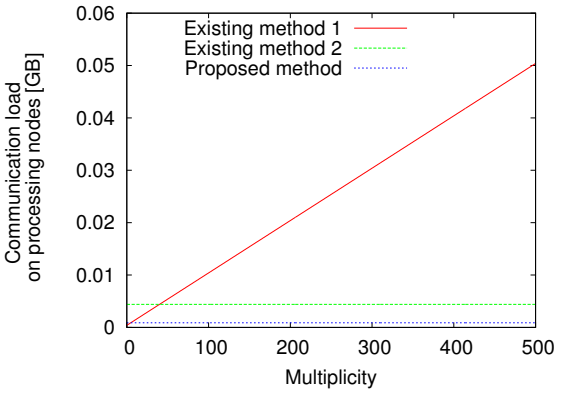

図 24　多重度に対する一段目の処理ノードにかかる通信負荷の比較

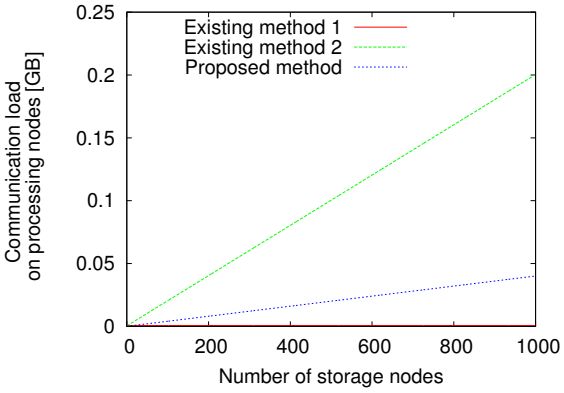

図 25　保管ノードの数に対する一段目の処理ノードにかかる通信負荷の比較

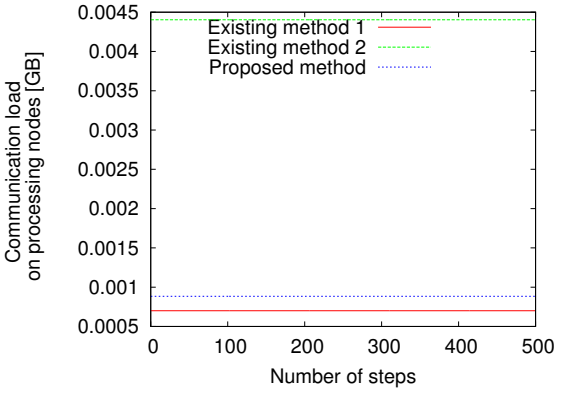

図 26　段数に対する一段目の処理ノードにかかる通信負荷の比較

する確率のすべての段にわたる累乗であることから，各段で多数決により正しい結果を採用する確率を高く保つ必要がある．段数を x，ある段で多数決により正しい結果を採用する確率を q としたときの，あるブロックにおいて正しくない結果を一度も採用しない確率 $C = q^x$ の算出例を以下に示す．

- $q = 0.99999999,\ x = 500$ の場合: $C \simeq 0.999995$.
- $q = 0.999,\ x = 500$ の場合: $C \simeq 0.606378$.

これらから，正しくない結果を一度も採用しない確率を高くするためには，多数決により正しい結果を採用する確

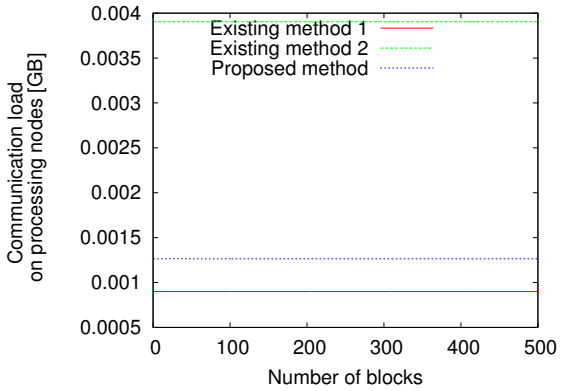

図 27　ブロック数に対する二段目以降の処理ノードにかかる通信負荷の比較

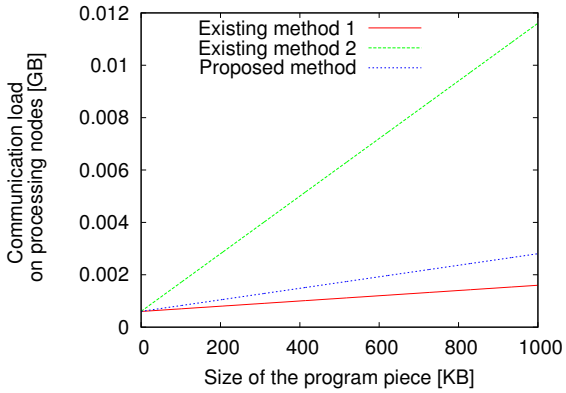

図 28　プログラム断片のサイズに対する二段目以降の処理ノードにかかる通信負荷の比較

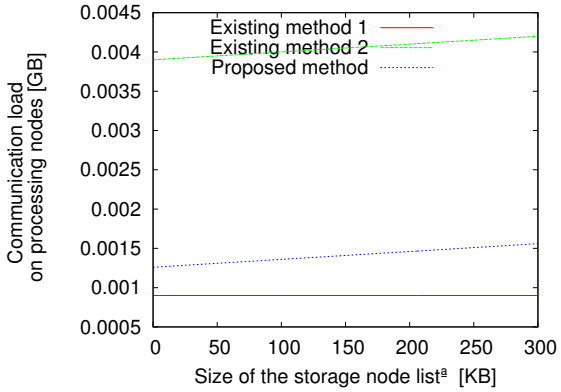

図 29　保管ノード表のサイズに対する二段目以降の処理ノードにかかる通信負荷の比較

率を高く保つ必要があることを例証できる．多数決により正しい結果を採用する確率を高く保つ方法としては，処理ノードが正しい結果を返す確率をあげる，多重度を高くするという二つが考えられる．前者は制御できず，それをあげることは現実には困難なため，多重度を高くする必要がある．多数決により正しい結果を採用する確率を 0.99999999 以上とするための，多重度および処理ノードが正しい結果を返す確率の組合せ例を，表 3 に示す．

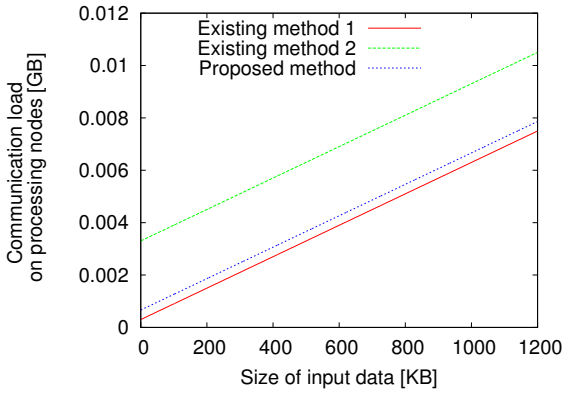

図 30　入力データのサイズに対する二段目以降の処理ノードにかかる通信負荷の比較

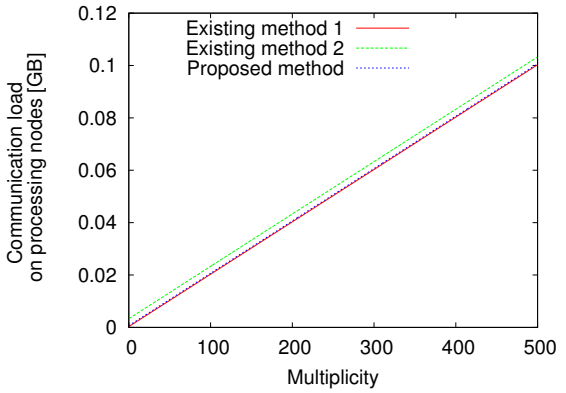

図 31　多重度に対する二段目以降の処理ノードにかかる通信負荷の比較

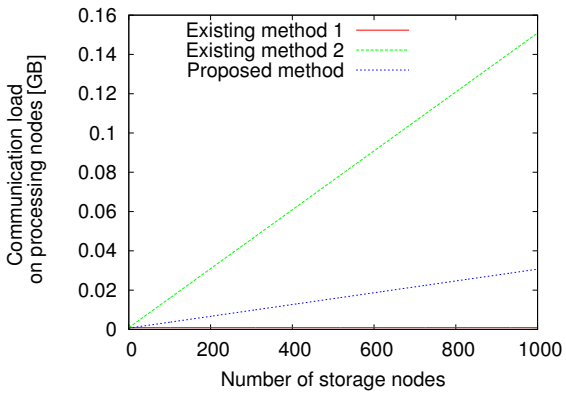

図 32　保管ノードの数に対する二段目以降の処理ノードにかかる通信負荷の比較

表 3 より，処理ノードが正しい結果を返す確率が 90%前後であると仮定すると，多数決により正しい結果を採用する確率が 0.99999999 となるためには，多重度を 30 程度にする必要があると推定できる．実際には，処理ノードが正しい結果を返す確率は不明であるため，グリッド全体としての信頼度をあげるためには，可能な限り多重度を高くすることが望まれる．

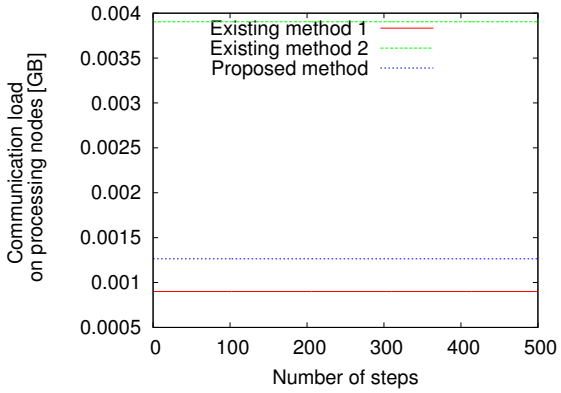

図 33　段数に対する二段目以降の処理ノードにかかる通信負荷の比較

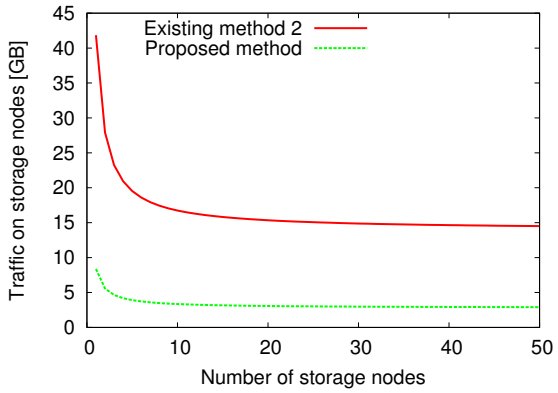

図 34　保管ノード数に対する保管ノードにかかる通信負荷の比較（多重度 30 の場合）

表 3　多重度と処理ノードが正しい結果を返す確率の関係

多重度	しきい値	処理ノードが正しい結果を返す確率
3	2	0.9978455653
5	3	0.9933036106
7	4	0.9862736727
9	5	0.9776045067
11	6	0.9680562773
13	7	0.9581611588
15	8	0.9482564999
17	9	0.9385441936
19	10	0.9291385768
39	20	0.8554201359
59	30	0.8085745706
79	40	0.7762124135
99	50	0.7522479629

5.5　まとめ

多重度に関する考察から，グリッド全体としての信頼度をあげるためには，多重度を高くする必要のあることがわかった．多重度を高くすると，$M > n/L$ が満たされやすくなり，(k, L, n) しきい値ランプ型秘密分散法を用いた通信負荷の軽減手法により，管理ノードにかかる通信負荷を軽減できる可能性が高くなる．

たとえば，図 10 より，多重度を 30 としたとき，保管ノードを用いない手法を用いた場合に管理ノードにかかる通信負荷は約 27.9[GB] となる．一方，多重度が同じとき，(k, L, n) しきい値ランプ型秘密分散法を用いた通信負荷の軽減手法による管理ノードにかかる通信負荷は，約 1.02[GB] である．このことから，多重度が高いほど，(k, L, n) しきい値ランプ型秘密分散法による通信負荷の軽減手法の効果をより見込めることがわかる．ここで，多重度の増加にともなう保管ノードの通信負荷の変化を考える．多重度を 30 としたときの，保管ノード数の増加に対する保管ノードにかかる通信負荷の，保管ノードを用いる二つの手法間での比較を図 34 に示す．

図 34 より，多重度を 30 にした場合でも，保管ノードの数を多く（30 以上）することで，保管ノードにかかる通信負荷を 3[GB] 程度に抑えられることがわかる．また，図 11 より，管理ノードにかかる通信負荷は，保管ノードの数を 417 以上にしない限り，既存手法 1 よりも提案手法のほうが効果を見込める．さらに，図 25, 32 より，処理ノードにかかる通信負荷は，保管ノードにかかる通信負荷に比べると軽いことがわかる．

以上のことから，多重度を高くする必要があり，かつ，通信負荷を減じたい場合，(k, L, n) しきい値ランプ型秘密分散法による通信負荷の軽減手法によって，管理ノードにかかる通信負荷を減らしつつ，保管ノード・処理ノードにかかる通信負荷をさほど重くないものにすることができる．

6.　おわりに

本稿は，信頼できない計算機を用いたグリッドコンピューティングであるエクスターナルグリッドを対象とし，信頼性を向上させるための手法である処理の多重化を用いると管理ノードに通信負荷が集中することを述べ，その対策として通信負荷の軽減手法を提案した．この手法は，(k, L, n) しきい値ランプ型秘密分散法を用いて，配布するデータを秘密化・分散化し，まずは複数の保管ノードへ配布し，それらから処理ノードへ配布するというものである．

各ノードにかかる通信負荷を定式化し，調査を行った結果，管理ノードの通信負荷については，ブロック数，プログラム断片のサイズ，段数，多重度の増加にともない，提案手法の効果を見込めることがわかった．一方，入力データのサイズ，保管ノードの数，保管ノード表のサイズについては，提案手法の効果を見込めないことがわかった．提案手法の効果が見込めない状況でも，$M > n/L$，つまり L を n/M より大きくすることで，入力データのサイズの増加にともない提案手法の効果が見込めることがわかった．保管ノードの通信負荷については，多重度の増加にともなって通信負荷が増すものの，その負荷は保管ノードを多くすることで軽減できることがわかった．このことから，多重度を高くする必要があり，かつ，通信負荷を減じたい場合

には，提案手法によって，管理ノードにかかる通信負荷を減らしつつ，保管ノード・処理ノードにかかる通信負荷をさほど重くないものにできることがわかった．

今後の課題として，処理ノードが秘密分散断片や秘密分散データから元のデータを復元する際にかかる計算時間を考慮することや，保管ノード・処理ノードにかかる通信負荷をどの程度まで許容できるかを視野に入れて評価することがあげられる．

謝辞

本研究の一部は JSPS 科研費（基盤研究 (C)）26330105 の助成を受けたものである．

参考文献

[1] 合田憲人，関口智嗣：グリッド技術入門，コロナ社 (2008).
[2] Himeda, K., Hirata, K., Higami, Y. and Kobayashi, S.: Consideration of Characteristics of Program for Concealing Purpose of Processing in Distributed Computing Systems, *Polish Journal of Environmental Studies*, Vol. 17, No. 4C, pp. 226–229 (2008).
[3] Sugimoto, K., Hirata, K., Higami, Y. and Kobayashi, S.: Multiplexing Scheme with Distributed Processing in External Grids, *Polish Journal of Environmental Studies*, Vol. 99, pp. 50–53 (2009).
[4] Miyaoka, H., Hirata, K., Higami, Y. and Kobayashi, S.: The Detection of Falsification with Check Codes in External Grid, *Polish Journal of Environmental Studies*, Vol. 17, No. 4C, pp. 294–298 (2008).
[5] Shamir, A.: How to share a secret, *Communication of the ACM*, Vol. 22, No. 11, pp. 612–613 (1979).
[6] 山本博資：(k,L,n) しきい値秘密分散システム，電気通信学会論文誌， Vol. J68-A, No. 9, pp. 945–952 (1985).