2005－EVA－15（4）
2005／11／22

グリッドコンピューティング環境における計算時間短縮のための大規模データ送信手法

野村成伸[†]　義久智樹[††]　金澤正憲[††]

近年，複数の計算機の余った計算リソースを活用するグリッドコンピューティングに対する注目が高まっている．グリッドコンピューティング環境では，一般に，実験施設内のサーバから発生する大規模なデータに対して複数の計算クライアントが異なる計算を並列に行うことで，計算完了までの時間を短縮している．しかし，クライアントの数が非常に多い場合，通信コストといったシステムの負荷が大きくなり，計算時間が長くなる可能性がある．そこで本稿では，計算時間短縮のため，放送型配信を用いた大規模データの送信手法を提案する．放送型配信では，サーバは各クライアントに同じデータをまとめて配信できるため，クライアント数が増加してもシステムの負荷はほとんど変わらないといった利点がある．クライアントは必要なデータが放送されるまで待つ必要があるが，評価の結果，データを分割して放送することで，計算時間を短縮できることが明らかになった．

A Delivering Method of Large Data for Computation Time Reduction on Grid Computing Environment

Shigenobu Nomura,[†] Tomoki Yoshihisa[††] and Masanori Kanazawa[††]

Recently, grid computing, which exploits extra computing resources in the network, has been attracted great attention. In grid computing environment, generally, by using several clients and calculating different equations against large data generated by a server, computation time is reduced. However, in the case where clients are too many, the system load becomes large and the computation time can be longer. In this paper, we propose a delivering method of large data for computation time reduction using broadcasting delivery. In broadcasting delivery, the server broadcasts the data to many clients concurrently. Accordingly, the system load does not change even if the number of clients get large. Although clients have to wait until their desired data is broadcast, our evaluations show that we can reduce computation time by dividing the data.

1. はじめに

近年，計算機の余った計算リソースを活用するグリッドコンピューティングに対する注目が高まっている[1),2)]．グリッドコンピューティング環境では，複数の計算機が連携して動作する必要があり，Globus[3)] やUNICORE[4)] といったグリッドコンピューティングのためのミドルウェアが開発されている．グリッドコンピューティング環境では，一般に，同じデータに対して複数の計算機が異なる計算を並列に行い，計算時間を短縮している．例えば，以下の状況が挙げられる．

- データベース
  大規模なデータベースに格納されたデータに対し，総和や平均といった統計値を複数の計算機で並列に計算する．
- ストリーミングデータ
  スーパーカミオカンデやスプリング 8 といった大規模な実験施設から発生するデータに対し，各研究グループが所望の計算を行う．

しかし，計算クライアントの数が多くなると，通信コストといったシステムの負荷が大きくなり，計算時間が長くなる可能性がある．そこで，本稿では計算時間短縮のため，放送型配信技術を用いた大規模データ送信手法を提案する．放送型配信では，サーバがデータを周期的に繰り返して放送することにより，各クライアントに同じデータをまとめて配信できる[5)~7)]．このため，クライアント数が増加してもシステムの負荷

† 京都大学大学院情報学研究科システム科学専攻
Graduate School of Informatics, Kyoto University
†† 京都大学学術情報メディアセンター
Academic Center for Computing and Media Studies, Kyoto University

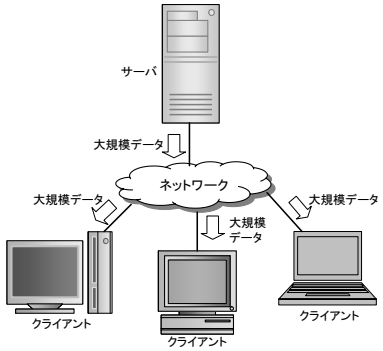

図 1 グリッドコンピューティング環境における放送型配信の例
Fig. 1 An example of broadcasting systems for grid computing environment.

はほとんど変わらないといった利点がある．クライアントはデータが放送されるまで待つ必要があるが，評価の結果，データを分割して放送することで，計算完了までの時間を短縮できることが明らかになった．

以下，第 2 章で放送型配信について説明した後，第 3 章で提案する送信手法について述べる．第 4 章でグリッドコンピューティング環境における放送型配信について評価し，第 5 章で考察を行う．最後に第 6 章でまとめる．

2. グリッドコンピューティング環境における放送型配信

放送型配信では，一般に，サーバは同じデータを繰り返して放送する．図 1 では，サーバに格納されている，もしくは，サーバから発生する大規模データを，ネットワークを介して繋がっているクライアントに放送している．サーバは，多くのクライアントにまとめてデータを配信できるが，クライアントは必要なデータが放送されるまで待つ必要がある．例えば，図 2 では「Broadcasting Data」はサーバが放送しているデータを示す．右に行く程時間が経過しており，サーバは，同じ「Large Data」を繰り返して放送している．このとき，時刻 t_1 にデータの受信を要求したクライアントは，時刻 t_2 に Large Data の受信を完了し，計算を開始できる．この例では，データの受信開始から受信完了までに d_1 かかっており，計算には d_2 かかっている．計算開始までの待ち時間は $t_2 - t_1$ となり，データの受信要求を出してから計算を完了するまで $t_2 - t_1 + d_2$ かかることになる．本稿ではこの計算時間を短縮するために，データを分割し，効率的にスケジューリングして放送する．

2.1 基本となるアイデア

本章では簡単な例を用いて，初めに大規模データを

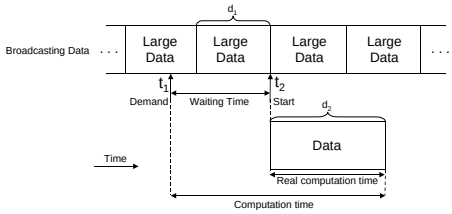

図 2 待ち時間の例
Fig. 2 An example of a waiting time.

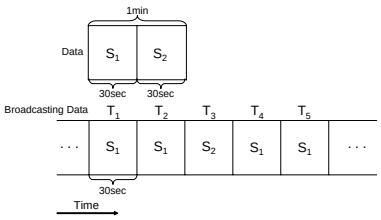

図 3 本手法のアイデアを説明するための例
Fig. 3 An example to explain the idea of our methods.

繰り返して放送する場合の平均計算時間を求め，次にデータを分割して放送する，本稿の基本となる考え方の簡単な例を示す．

まず，放送に 1 分（$d_1 = 1$ 分）かかり，計算に 2 分（$d_2 = 2$ 分）かかるデータの場合を考える．これを図 2 のように繰り返し放送する．この場合，クライアントがランダムに受信要求を出すと，平均計算時間は 3 分 30 秒となる．

次に，提案手法のアイデアを説明する．先程と同じデータを，30 秒ずつの 2 つのデータに分割した場合を考える（図 3）．このとき，計算に要する時間はそれぞれ 1 分ずつとなる．分割したデータの前半部分を S_1，後半部分を S_2 とし，クライアントは S_1 に対して計算を行った後，S_2 に対して計算を行うとする．サーバはデータの前半部分を頻繁に放送するように，$S_1S_1S_2$ の順に繰り返し放送する．S_1，S_2 の放送にそれぞれ 30 秒ずつかかるので，$S_1S_1S_2$ の放送周期は，1 分 30 秒になる．ここで，最初の S_1 の放送区間を T_1，次の S_1 の放送区間を T_2，最後の S_2 の放送区間を T_3 とする．平均待ち時間は，以下のような場合分けにより求めることができる．

- T_1 内でクライアントが計算を開始した場合
  クライアントは T_2 で放送される S_1 を受信し，S_1 に対する計算を行いながら T_3 で放送される S_2 を受信できる．つまり，T_2 で S_1 を受信完了すると同時に計算を開始できる．この場合の計算時間は 2 分 45 秒となる．
- T_2 内でクライアントが計算を開始した場合

クライアントは T_4 で S_1 の受信完了と同時に計算を開始できる．この場合の計算時間は 3 分 15 秒となる．

- T_3 内でクライアントが計算を開始した場合
  クライアントは T_4 で放送される S_1 を受信し，受信完了すると同時に計算を開始できる．この場合の計算時間は 2 分 45 秒となる．

以上より，平均計算時間は 2 分 55 秒となり，前述の分割しない方法と比べると待ち時間は 17 %短縮されている．

2.2 クライアントの計算順序

グリッドコンピューティング環境では，各クライアントの計算能力は大きく異なることが多い．しかし本稿では，簡単のため，全クライアントのデータの計算にかかる時間を D とし，これを実計算時間と呼ぶ．計算に必要なデータを放送するのにかかる時間を D' で表し，計算倍率 a を

$$a = \frac{D}{D'} \tag{1}$$

と定義する．例えば，先程の例のように計算に 2 分かかり，そのデータを放送するのに 1 分かかるデータの場合，$a = 2$ になる．

データを分割して放送する場合，クライアントの計算の順序として，シーケンシャルなデータ処理と FIFO 処理の 2 通りを考える．以下に説明する．

2.2.1 シーケンシャル

シーケンシャルな処理では，データの初めから順番に計算を行う．例えば，$S_1S_1S_2$ の順に放送し，$a = 1.5$ の場合を図 4 に示す．図 4 で，時刻 t_1 でデータの受信を要求する場合を考える．クライアントは，次に放送される S_2 から受信を開始することになる．S_2 を受信完了すると，S_2 を保持し，次の S_1 の受信を開始する．S_1 の受信完了と同時に t_2 で計算を開始し，データの順番である S_1，S_2 の順に計算する．このとき，データの受信を要求した t_1 から，計算を開始する t_2 までが待ち時間，S_1 と S_2 の両データに対する計算が完了する t_3 までが計算時間，$t_3 - t_2$ が実計算時間となる．

2.2.2 FIFO

FIFO 処理では，受信完了したデータから順に計算を行う．つまり，要求を出してから，最初のセグメントを受信完了すると同時に計算を開始することになる．例えば，$S_1S_1S_2$ の順に放送し，$a = 1.5$ の場合を図 5 に示す．図 5 で，時刻 t_1 でデータの受信を要求する場合を考える．時刻 t_2 で S_2 を受信完了すると，S_1 の受信完了を待たずに S_2 に対する計算を開始する．そして，S_2 に対する計算を行っている間に S_1 を受信する．S_2 の計算終了後，S_1 に対する計算を行う．このとき，データの受信を要求した t_1 から，計算を開始する t_2 までが待ち時間，S_1 と S_2 の両データに対する計算が完了する t_3 までが計算時間，$t_3 - t_2$ が実計算時間となる．

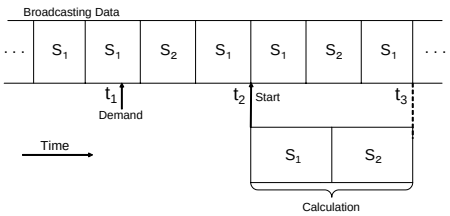

図 4 シーケンシャルな処理 ($a = 1.5$)
Fig. 4 A sequential processing($a = 1.5$).

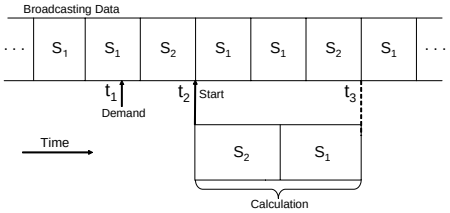

図 5 FIFO 処理 ($a = 1.5$)
Fig. 5 A FIFO processing($a = 1.5$).

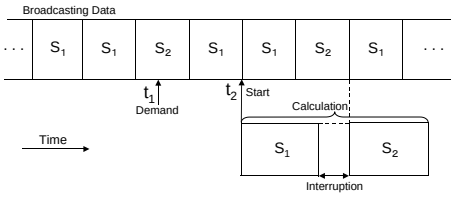

図 6 途切れの例
Fig. 6 An example of an interruption.

2.3 計算途切れ時間

シーケンシャルな処理の場合を例に，図 6 を用いて計算途切れ時間について説明する．クライアントが，時刻 t_1 でデータの受信を要求する場合を考える．クライアントは，次に放送される S_1 から受信を開始することになる．t_2 で S_1 を受信完了すると，S_1 に対する計算を開始する．しかし，S_1 に対する計算を完了するのは S_2 を受信している途中なので，S_2 を受信完了するまで S_2 に対する計算は行えない．このように，あるデータに対する計算が終わってから，次のデータの受信が完了するのを待ち，計算を開始するまでの時間を計算途切れ時間と呼ぶ．

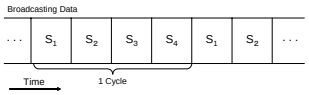

図 7 単純法の例
Fig. 7 An example for the simple method.

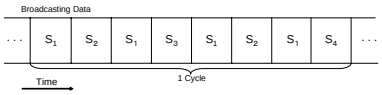

図 9 セグメント挿入法の例
Fig. 9 An example for the segment insertion method.

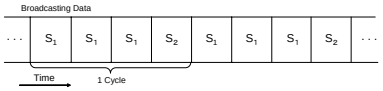

図 8 二等分法の例
Fig. 8 An example for the hemisection method.

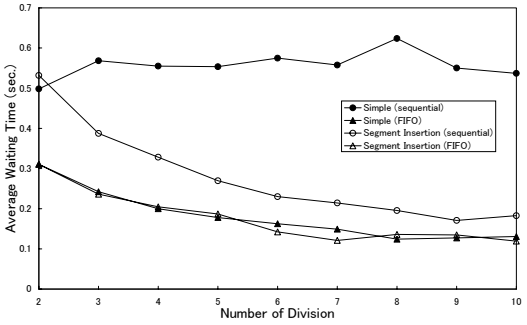

図 10 分割数に対する平均待ち時間
Fig. 10 The average waiting time and number of division.

3. 送信手法

以下，分割したデータのスケジューリング手法を説明する.「単純法」,「二等分法」,「セグメント挿入法」について説明する.

3.1 単純法

単純法では，データを N 等分（$N = 1, 2, ...$）し，S_1 から S_N までを繰り返し放送する．4 等分して単純法で放送する場合の例を図 7 に示す.

3.2 二等分法

二等分法では，データを二等分し，前半を S_1，後半を S_2 とする．本手法では，S_1 を M 回繰り返し放送する．$M = 1$ の場合は前節の単純法となる．例えば，$M = 2$ の場合は $S_1S_1S_2$ の順に繰り返し放送され，$M = 3$ の場合は $S_1S_1S_1S_2$ の順に繰り返し放送される．$M = 3$ の場合の例を図 8 に示す.

3.3 セグメント挿入法

セグメント挿入法では，データを N 等分（$N = 2, 3, ...$）し，以下の手順に従ってスケジューリングする.

(1) データを，セグメント $S_1, ..., S_N$ に N 等分（$N = 2, 3, ...$）する.
(2) セグメント S_N を n_N 個並べる.
(3) $i = N - 1$ として，並べたすべてのデータの左側に S_{N-1} を各々 n_{N-1} 個並べる.
(4) i を 1 つずつ減らしていき，これを $i = 1$ まで繰り返す.

本稿では，簡単のため，n_i（$i = 1, ..., N$）は全て 1 で実験を行った．$N = 4$，$n_i = 1$（$i = 1, ..., 4$）の場合の例を図 9 に示す.

4. 評価

計算機シュミレータを作成し，評価を行った．評価には，放送に 1 秒を費やすデータを用い，クライアントの計算時間を変更することで、a を変化させた.

4.1 単純法とセグメント挿入法の比較

4.1.1 平均待ち時間

単純法およびセグメント挿入法の平均待ち時間を図 10 に示す．縦軸は平均待ち時間，横軸は分割数とした．グラフは，a を 1 から 10 まで変化させたときの平均である.

シーケンシャルな処理について見ると，このグラフより，セグメント挿入法は単純法に比べ平均待ち時間が大きく短縮されていることがわかる．例えば，分割数が 6 のときの平均待ち時間は，単純法では 0.62 秒でセグメント挿入法では 0.19 秒である．これは，セグメント挿入法では，分割数を大きくすると始めのデータを放送する割合が大きくなり，計算を開始する機会が増えたためである.

FIFO 処理の場合，このグラフより，単純法とセグメント挿入法では大きな差が無いことがわかる．これは，FIFO 処理が受信完了したデータから計算を行うためであり，分割数が同じであれば，平均待ち時間は等しくなると考えられる.

4.1.2 シーケンシャルな処理における平均計算時間

シーケンシャルな処理における，単純法およびセグメント挿入法の平均計算時間を図 11 に示す．縦軸は平均計算時間，横軸は分割数とした．このグラフより，a の値が大きくなると，セグメント挿入法は単純法に比べ計算時間が短縮されていることがわかる．例えば分割数が 5 のときで比較すると，$a = 2$ のときは両手法とも計算時間はほぼ同じであるが，$a = 5$ のときは 0.13 秒，$a = 10$ のときは 0.35 秒，セグメント挿入法

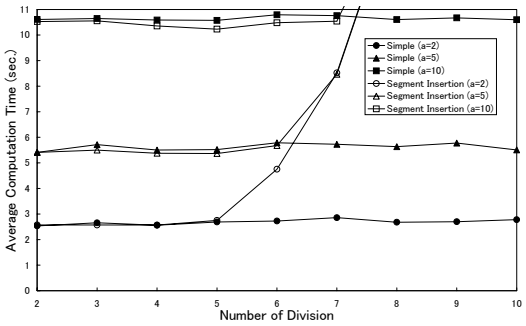

図 11 分割数に対する平均計算時間（シーケンシャル）
Fig. 11 The average computation time and number of division(sequential).

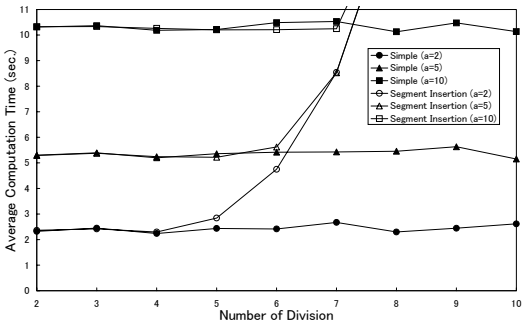

図 13 分割数に対する平均計算時間（FIFO）
Fig. 13 The average computation time and number of division(FIFO).

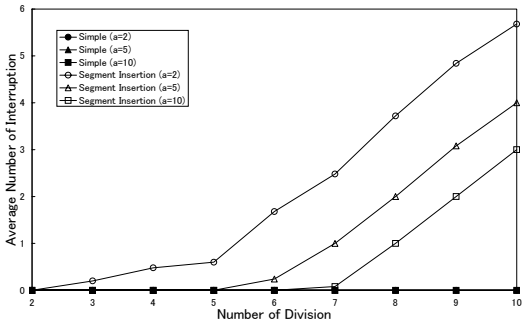

図 12 分割数に対する平均途切れ回数（シーケンシャル）
Fig. 12 An average number of interruption and number of division(sequential).

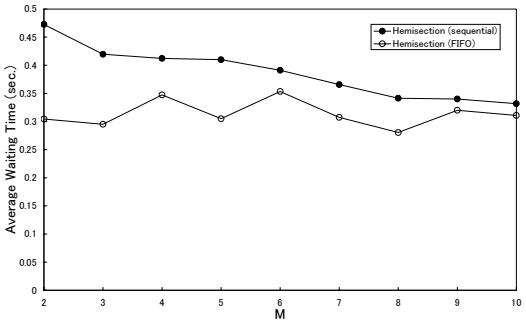

図 14 M に対する平均待ち時間
Fig. 14 The average wawiting time and M.

は単純法に比べ計算時間が短縮されている．しかし，分割数が大きすぎると，急激に計算時間が増大している．これは，図 12 よりわかるように，途切れが原因であると考えられる．図 12 において，セグメント挿入法では分割数が 5 あたりから平均途切れ回数が急激に多くなっている．これは，分割数を大きくしたことで始めのデータを放送する割合が増え，他のデータを受信していない状態で計算を開始するためである．

4.1.3 FIFO 処理における平均計算時間

FIFO 処理における，単純法およびセグメント挿入法の平均計算時間を図 13 に示す．縦軸は平均計算時間，横軸は分割数とした．このグラフより，FIFO 処理ではどの a においても計算時間は短縮されていないことがわかる．しかし，シーケンシャルな処理の場合と同じく，分割数が大きすぎると，急激に計算時間が増大している．これもシーケンシャルな処理の場合と同様で，途切れが原因であると考えられる．

4.2 二 等 分 法

次に，二等分法の評価を行う．

4.2.1 平均待ち時間

図 14 に，二等分法での M に対する平均待ち時間を示す．縦軸は平均待ち時間，横軸は M とした．グラフは，a を 1 から 10 まで変化させたときの平均である．このグラフより，FIFO 処理では M に対し平均待ち時間の変化は見られなかったが，シーケンシャルな処理では，M を大きくすると平均待ち時間は短縮されていることがわかる．これは，M を大きくすることで 1 周期中の S_1 の割合が大きくなり，計算を開始する機会が増えたためである．

4.3 平均計算時間

図 15 に，二等分法での a に対する平均計算時間を示す．縦軸は平均計算時間，横軸は a とした．グラフより，$a = 2.5$ では $M = 2$ で，$a = 3.5$ では $M = 3$ で，$a = 4.5$ では $M = 4$ で計算時間が最小になっている．これより，a の値によって計算時間が最小となる M が異なることがわかる．これは，M の値によって，途切れが発生せずに，最適な計算を行うことができる a の値が異なるためである．

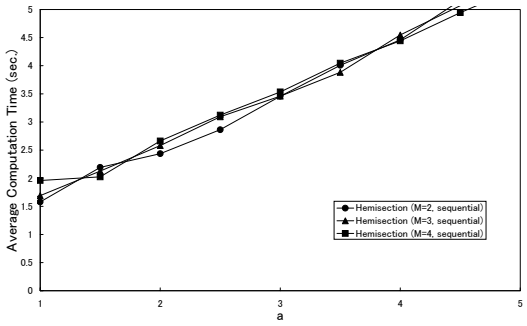

図 15 a に対する平均計算時間
Fig. 15 The average computation time and a.

5. 考　　察

5.1 計算時間の短縮

実計算時間は短縮されていないが，データを分割することで待ち時間を短縮でき，計算時間の短縮に繋がった．例えば，図 10 に示したように，データを分割することで，シーケンシャルな処理では待ち時間が短縮されたが，FIFO 処理では待ち時間は短縮されなかった．これより，シーケンシャルな処理では計算時間を短縮でき，FIFO 処理ではあまり短縮できないことがわかる．

5.2 単純法とセグメント挿入法

図 10 でシーケンシャル方式を見ると，単純法に対しセグメント挿入法では平均待ち時間は大きく短縮されていることがわかる．これは，分割数を増やすことにより始めのデータを放送する割合が増え，計算を開始する機会が増えたからである．例えば，データを 4 等分して単純法で放送している図 7 と，$N = 4$，$n_i = 1$ $(i = 1, ..., 4)$ でセグメント挿入法で放送している図 9 を比べると，単純法では 1 周期の 4 セグメント中に S_1 は 1 つしかないが，セグメント挿入法では 1 周期の 8 セグメント中に S_1 は 4 つあり，S_1 を受信する機会は多い．上記の結果が影響し，図 11 の分割数が 3，4，5 あたりで平均計算時間が短縮されていることがわかる．しかし，分割数が 5，6 あたりを超えてくると，平均計算時間は急激に増大している．これは途切れによるもので，図 12 を見ると分割数が 5 あたりから平均途切れ回数が増えていることがわかる．

図 10 で FIFO 処理を見ると，単純法とセグメント挿入法で平均待ち時間の変化は見られなかった．これは，FIFO 処理では受信したデータの順に計算を行うので，分割数が同じであれば，待ち時間は等しいからと考えられる．上記の結果より，図 13 の平均計算時間でも，単純法とセグメント挿入法で変化は見られなかった．分割数が 5，6 あたりを超えてくると，平均計算時間は急激に増大しているのはシーケンシャルな処理の場合と同じく途切れによるものと考えられる．

5.3 二 等 分 法

図 14 より，FIFO 処理では分割数に対し平均待ち時間の変化は見られなかったが，シーケンシャルな処理では，分割数を大きくすると平均待ち時間は短縮されていることがわかる．これは，S_1 の放送の割合を増やすことにより平均待ち時間が短縮されたためと考えられる．例えば，$S_1S_1S_2$ の $M = 2$ のスケジュールより，$S_1S_1S_1S_1S_1S_2$ の $M = 5$ のスケジュールの方が，S_1 の割合が多く，計算を開始する機会が多いためである．

6. ま と め

本稿では，計算時間短縮のため，放送型配信を用いた大規模データの送信手法を提案した．放送型配信では，サーバは各クライアントに同じデータをまとめて配信できるため，クライアント数が増加してもシステムの負荷はほとんど変わらないといった利点がある．評価の結果，単純に放送する単純法と比べて，データを分割して放送する二等分法やセグメント挿入法では，計算時間を短縮できることが明らかになった．

今後の課題として，データに複雑な順序がある場合や，さらに計算時間や途切れを減少させるスケジューリング手法の考案を考えている．

参 考 文 献

1) Terence, H.: GridCast A Service Architecture for the Broadcasting Media,UK sScience All Hands Meeting(2004)
2) 超高速コンピュータ網形成プロジェクト (NAREGI): 国立情報学研究所開発推進拠点, http://www.naregi.org/index.html
3) The Globus Alliance(1998), http://www.globus.org/
4) UNICORE Forum e. V. , http://www.unicore.org/
5) Janakiraman, R. and Waldvogel, M.: Fuzzycast: Efficient Video-on-Demand over Multicast, *Proc. IEEE Infocom*(2002)
6) 義久智樹, 塚本昌彦, 西尾章治郎: 連続メディアデータ放送における待ち時間短縮のための分割放送方式, 情報処理学会論文誌, Vol. 44, No. 6, pp. 1558-1569(2003)
7) 義久智樹, 塚本昌彦, 西尾章治郎: データの分割に関する制約を考慮した連続メディアデータ放送におけるスケジューリング手法, 情報処理学会論文誌, Vol. 44, No. SIG3(TOD17), pp. 33-42(2003)