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

2003－DPS－115　(8)
2003／11／14

市場モデルに基づく計算機資源配分における分散実行系の提案

玉井 森彦　柴田 直樹　安本 慶一　伊藤 実

あらまし　本稿では，グリッドコンピューティング環境下で，各種計算機資源を売買するネットワーク上の市場を想定し，そこでの価格調停者（市場サーバ）を複数の計算機で分散実行する方式を提案する．提案方式では，注文が増えることによるサーバの負荷を抑えるため，注文の各パラメタ値を，例えば，CPU パワーが，100-200 MIPS，メモリ量が 50-100M バイトなどのように，一定値ごとに格子状に区切って出来る部分空間を考え，各部分空間を異なった市場サーバに割り当てる．各市場サーバは，自分に割り当てられた部分空間に入る注文のマッチングを行う．あるサーバに保存されている反対注文とマッチングさせるだけで，全ての注文とマッチングさせるのと同じ結果が得られるよう，各部分空間について，その部分空間の各パラメタ値よりも高い範囲の全ての注文の中で最も安い（高い）価格を提示している注文を自動的にその部分空間中に含めるためのアルゴリズムを考案した．

Distributed Markets for Market-based Resource Allocation

Morihiko Tamai　Naoki Shibata　Keiichi Yasumoto　Minoru Ito

Abstract　In this paper, we assume a market for buying/selling resources on network in grid computing environments and propose a method for realizing a resource coordinator (a market server) as a several distributed servers. In the proposed method, in order to regulate load on each server, we consider subspace of orders by partitioning each parameter like CPU power with 100 to 200 MIPS and memory amount with 50 to 100MB, and assign each subspace to a server. Each server handles orders which are included in the subspace. In general, when a new order comes, in order to find the opposite order with the lowest (highest) price, we have to search all regestered orders in the assigned subspace whose parameter values are equal to or larger than the new order. We have developed an algorithm which automatically stores in each subspace the order with the lowest (highest) price in all subspaces whose parameter values are larger than the subspace, so that each new order can find the best opposite order by searching only one server.

1　はじめに

近年の PC の高性能化および低価格化，ならびにインターネットへのブロードバンド接続ユーザの普及により，複雑かつ大規模な処理をインターネット上の多数の PC により分散実行するためのグリッドコンピューティングに関する研究が，最近注目を集めている．これらグリッドコンピューティングでは，インターネット上の一般ユーザから計算機資源を調達する必要があり，資源を提供するユーザへのインセンティブをどのように設定するかが，重要な問題となっている．

現在，PC を用いたグリッドコンピューティングの例として，SETI@home[1] や distributed.net [2] などのように，公共の目的のために，計算機が遊休状態にある時の CPU パワーをユーザが無償で提供する方式や，cell computing [3] のように，特定のサービス提供会社が，商用目的でユーザから集めた遊休計算機資源を顧客に有料で提供し，資源提供者には，何かしらの報償により還元を行う方式が存在する．しかし，これらの方式は，ユーザがある特定の事業者に資源を提供するだけの一方向のサービスであり，ユーザ間で自由に資源を融通し合いたいといった用途には向いていない．

上記を解決する方法として，分散環境での，市場原理に基づいた計算機資源の共有に関する研究が幾つか行われている [4, 5, 6, 7]．文献 [4] では，モバイルエージェントシステムにおいて，入札（sealed-bid second-

奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology

price auction）方式を利用して，資源管理を行う方法が提案されている．文献 [5] では，インターネット上の計算機を対象に CPU 時間の売買を double auction 方式により行う仮想市場 POPCORN が提案されている．また，文献 [6] では，グリッドコンピューティング環境において，需要と供給のバランスに基づいて，各資源やコンポーネントの価格決定を行うためのフレームワークが提案されている．さらに，文献 [7] は，市場サーバを用いて，Java および Web ベースのシステムから，グリッド計算環境における資源の調達を可能にするフレームワークおよびプロトタイプシステムを提案している．これら既存の提案では，市場原理に基づいた資源管理法のフレームワークを提案しているものがほとんどで，市場原理に基づく資源の価格の決定に単一の調停者あるいは市場サーバの存在を仮定しており，インターネット上の不特定多数の計算機を対象として機能させるための，市場サーバの分散実装方式については考慮されていない．

本稿では，グリッドコンピューティング環境下で，市場原理に基づき各種計算機資源を売買する市場を想定し，そこでの価格調停者（市場サーバ）を複数の計算機で分散実行する方式を提案する．

本稿で対象とする市場モデルでは，CPU パワー，メモリ量，占有時間，通信帯域，遅延時間，価格などの複数パラメタからなる売買注文を受け，各注文に対し，必要な資源量を満たし（買注文の場合，各項目について指定した値以上，ただし，遅延時間，価格については指定値以下），かつ，ユーザにとって最も得となるような（条件を満たす反対注文の中で最も安い，あるいは高い）反対注文を見つける市場サーバの存在を仮定する．この市場サーバを分散で実装することが本稿の目的である．

本稿では，市場サーバを分散する上で，(1) 各サーバで扱う注文数を一定値以下に抑えることができること，(2) マーケットが分散されないこと，すなわち，マッチングの結果が，単一サーバでマッチングを行った場合と同じであること，を目標とする．これ以外にも，P2P 環境において，特定の事業者の介在なく計算機資源の売買を実現するには，サーバを固定するのでなく，P2P ネットワークに参加しているユーザの誰かがサーバとなることができること，が必要となる．これを実現するためには，P2P ネットワークに特有の問題，すなわち，サーバノードが事前の申告無く任意のタイミングでネットワークから離脱する際のバックアップ方式，および悪意のあるユーザからの被害を防止するためのセキュリティを実現する必要があり，これらの問題に対しては，本稿の対象外とする．

提案方式では，注文が増えることによるサーバの負荷を抑えるため，資源の各パラメタ値を一定値ごと格子状に区切って出来る部分空間を考え，部分空間ごとに別のサーバに分散配置する方法を採用する（この際，複数の部分空間を一つのサーバに対応させても良い）．各部分空間は，例えば，CPU パワーが，100-200 MIPS，メモリ量が 50-100M バイト，通信帯域が 200-300Kbps などのように区切られており，この範囲に入って来た売注文*は，入り口サーバのインデックスを見て，この部分空間の担当サーバに転送され保管される．一方，買注文がこの範囲に入ってきた時には，このサーバに保管された売注文とマッチングが行われ，条件を満たし，かつ，最も安いものが選ばれる．ただし，この方法では，この部分空間の範囲外で，より豊かな資源をより安い価格で提示している売注文（例えば，CPU パワー 300MIPS，メモリ 80M バイト，帯域 1Mbps）とのマッチングができない．この問題を解決する最も単純な方法は，該当部分空間よりも高い範囲の全ての部分空間の売注文とマッチングを行うことであるが，マッチングのコストが高くなり，分散実行の意義が薄くなる．提案方式では，各部分空間 A について，A の各パラメタ値よりも高い範囲の全ての部分空間の中で最も安い価格を提示している売注文を A 中に含めることとした．これにより，部分空間 A の範囲に入る買注文に対して，A 中の売注文とマッチングさせるだけで，最適な結果が得られる．以上を実現するため，各部分空間を担当するサーバは，売注文を受け取るたびに，部分空間中の最安値売注文と比較し，最安値を更新する場合には，パラメタ値が減少する方向に隣接する部分空間のサーバにその売注文の存在を通知するための機構を考案した．

2 計算機資源市場

2.1 概要

以下では，提案する計算機資源市場の概要について述べる．提案する計算機資源市場を用いて，ユーザはグリッドコンピューティング環境下等において，計算機資源を互いに融通しあう．自分の持つ計算機資源に余裕のあるユーザは，市場に自分の持つ計算機資源を登録する（売注文を出す）．売注文は，自分の持つ計算機資源を表す，複数のパラメータ値と，価格からな

*以下，各部分空間へ売注文を格納する場合を想定して話を進めるが，買注文の場合も同じである．

る．複数のパラメータの集合は，例えば {CPU パワー，メモリ量，計算機の占有時間，通信帯域，通信の遅延時間} といったものが考えられる．市場に売注文が十分ある状態で，計算機資源が欲しいユーザが買注文を市場に出す．買注文は先ほどと同じ組み合わせの，複数のパラメータ値からなる．市場サーバは，自分の保持する売注文の中にパラメータ値がいずれも買注文のパラメータ値以上であるような売注文があるかどうか調べ，ある場合はその売注文の中で最も価格の安いものを買い注文を出したユーザに返す（マッチングの定義）．また，売注文を出したユーザは，いつでもその注文をサーバから削除することができる．

2.2 詳細

以下では，提案する計算機資源市場において使用される市場サーバの詳細な仕様について述べる．

最初に，売注文を格納する変数の型である， Order 型を定義する．

```
struct Order {
   // 各資源の売却量を表すベクトル
   real v[1..N];
   real price;   // 価格
   int id;       // 注文に固有の ID
   // クライアントの IP アドレスなど
   クライアントの情報 info;
};
```

関数　*AddSellOrder*
入力　登録された売注文の集合 S, 売注文 x
出力　売注文の集合 S', 売注文の番号 idx
入出力の満たす性質　S に含まれる任意の異なる 2 つの要素に対して，メンバ変数 id が異なる．S に含まれる任意の要素に対して，メンバ変数 id と idx が異なる．y を以下のように定義する：y のメンバ変数は，id を除いて x と同じ．$y.id$ は idx に等しい．$S' = S \bigcup \{y\}$.

関数　*DeleteSellOrder*
入力　登録された売注文の集合 S, 売注文 x
出力　売注文の集合 S'
入出力の満たす性質　S に含まれる任意の異なる 2 つの要素に対して，メンバ変数 id が異なる．$x \in S$. $S' = S - \{x\}$.

関数　*MatchOrder*
入力　登録された売注文の集合 S, 買注文 x
出力　売注文 y，ブール値 b

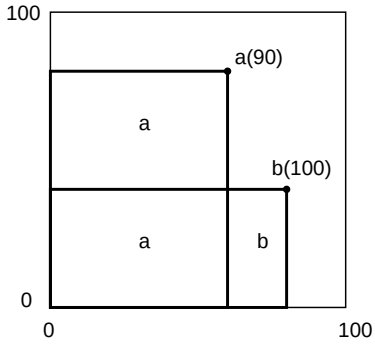

図 1: 市場サーバに保持される 2 つの売注文

入出力の満たす性質　S に含まれる任意の異なる 2 つの要素に対して，メンバ変数 id が異なる．S'' を以下のように定義する：$S'' = \{z \in S \mid \forall_{i \in \{1,2,\ldots,N\}}, z.v[i] \geq x.v[i]\}$. $S'' = \emptyset$ ならば $b = false$, $y =$ 不定．$S'' \neq \emptyset$ ならば $b = true$, y は S'' の要素のうち，$price$ メンバの値が最小のもの．

3 市場サーバの分散実装

3.1 概要

以下では，各計算機資源を 2 つのパラメータで表す場合について例を挙げながら，提案する分散アルゴリズムについて説明する．

市場サーバ群に 2 つの売注文 a, b が保持されているとする．$a.\vec{v} = (60, 80)$, $a.price = 90$, $b.\vec{v} = (80, 40)$, $b.price = 100$ とする．図 1 は，各売注文がマッチングする買注文の範囲を図示したものであり，各矩形領域内の a, b という記号は，その領域内に入る買注文がマッチする売注文を表している．

以下では，注文のマッチング等を 4 つの市場サーバ m_1, m_2, m_3, m_4 で分散処理する例について見ていく．また，これら 4 つの市場サーバの他に，1 つの入り口サーバが存在するとする．各座標軸が計算機資源の 2 つのパラメータを表すような，2 次元ベクトル空間を，図 2 に示したように 4 つに分割し，それぞれの部分空間を各市場サーバに割り当てる．市場サーバ m_j に割り当てられた部分空間を W_j とする．

注文の登録

新たな注文 x は，まず入口サーバへ送信される．入口サーバは，$x.\vec{v} \in W_j$ となるような W_j を割り当てられた市場サーバへ x を転送する．

売注文 d が W_1 に含まれる場合を考える（図 3 参照）．売注文 d がマッチする買注文の範囲は W_1, W_2, W_3, W_4 に及ぶ．そこで，これらに部分空間が割り当てられた

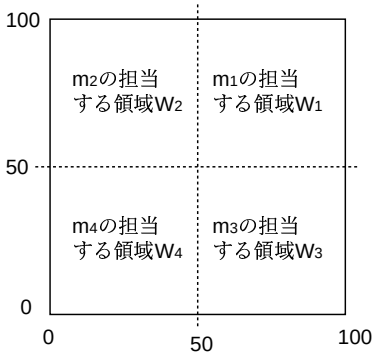

図 2: 4 つの市場サーバの担当する領域

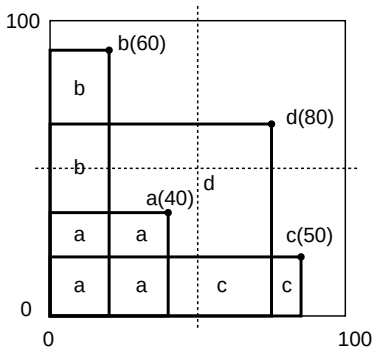

図 3: 市場サーバに保持される売注文 a, b, c, d

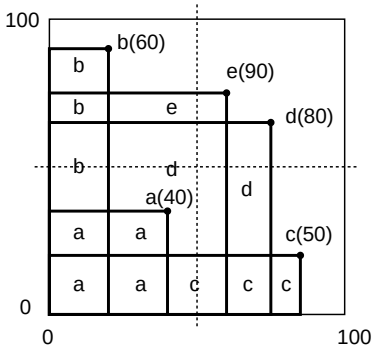

図 4: 新たに登録された売注文 e

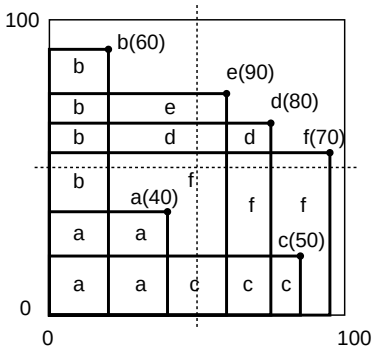

図 5: 新たに登録された売注文 f

市場サーバ全てで売注文 d を保持する．

各サーバに対する**下流隣接サーバ**を，パラメータ値が低い方向に隣接した領域を割り当てられたサーバとする (詳細な定義は後述する)．m_1, m_2, m_3, m_4 の下流隣接サーバはそれぞれ $\{m_2, m_3\}, \{m_4\}, \{m_4\}, \{\}$ となる．各市場サーバは，下流隣接する市場サーバへメッセージを送信する際のアドレスを保持している．例えば図 2 では，$m_1.Neighbor[1]$ に m_2 のアドレス，$m_1.Neighbor[2]$ に m_3 のアドレスを格納する．m_i が新たに保持する注文 x の領域が，m_i に下流隣接する市場サーバにも及ぶ場合は，x の複製を下流隣接する市場サーバにも保持させる．例えば，前述の売り注文 d は，注文の登録時にサーバ m_1 から m_2, m_3 に送られ，m_2, m_3 から m_4 に送られる．

次に，図 4 に示した売注文 e をサーバ群に登録することを考える．この場合，注文 e がマッチする買注文の範囲は全市場サーバに及ぶが，$e.price > d.price$ であるため，注文 d と重なる領域に入る買注文に関しては，注文 d が優先的にマッチする．従って，m_3, m_4 のように，既に注文 d を保持しており，かつ，自分の担当する領域において，注文 e の領域が注文 d と完全に重複する場合には，注文 e を保持しないこととする．これにより，W_3, W_4 が保持すべき売り注文の数を抑えることができる．

次に，図 5 に示した売り注文 f をサーバ群に登録することを考える．この場合，注文 f がマッチする買い注文の範囲は，全市場サーバにおよび，それに伴って，m_3, m_4 では注文 d を保持しておく必要がなくなる．提案方式では，m_3, m_4 から注文 d を削除することで，各市場サーバが保持すべき注文の数を抑える．

注文の削除

次に，市場サーバ m_i が保持する売注文 x が，買注文とマッチした場合に，注文 x を売注文の集合 $\mathcal{S}_i$ から削除することを考える．このとき，単純に x を $\mathcal{S}_i$ から削除するだけでなく，他の市場サーバに保持されている x の複製も削除する必要がある．

提案手法では，x を削除する場合には，$x.\vec{v} \in W_j$ となる市場サーバ m_j に，x の削除を要求するメッセージを送信する．次に，m_j に下流隣接する各市場サーバへも x を削除するためのメッセージを送信する．

また，売注文を $\mathcal{S}_i$ から削除することにより，$\mathcal{S}_i$ のいずれかの要素を，下流隣接市場サーバに保持させる必要が生じる場合がある．例えば，図 5 において，注文 d, f を削除すると，図 6 となる．この場合には，m_3, m_4 に注文 e の複製を保持させる必要が生じる．そのため，ある注文を削除した後の注文の集合の要素

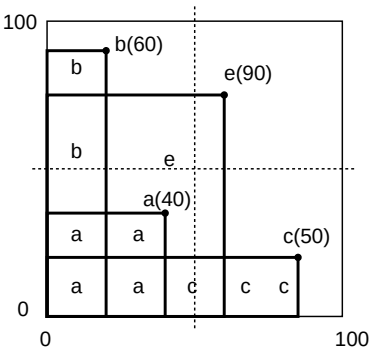

図 6: 図 5 から注文 d, f を削除した後

中に，下流隣接市場サーバが保持すべき注文がある場合には，その注文を追加するメッセージを送信する．

3.2 詳細

族 $P = \{W_1, ..., W_n\}$ を，N 次元ベクトル空間 W の分割とする．P の各要素は，ある値 $\alpha_{i,1}, ..., \alpha_{i,N}, \beta_{i,1}, ..., \beta_{i,N}$ に対して，$W_i = \{(x_1, ..., x_n) | \alpha_{i,1} \leq x_1 < \beta_{i,1} \wedge ... \wedge \alpha_{i,N} \leq x_N < \beta_{i,N}\}$ とする．

定義 $P = \{\vec{p_0}, \cdots, \vec{p_k}\}$ を d 次元ユークリッド空間内の有限点集合とする．$x = \Sigma_{i=0}^{k} \lambda_i \vec{p_i}$ かつ $\Sigma_{i=0}^{k} \lambda_i = 1$ であるとき，x は P の**アフィン結合**であるという．P のアフィン結合の集合を**アフィン空間**と呼ぶ．$p_i \in P$ が $P - \{p_i\}$ のアフィン結合になるような p_i が存在しないとき，P はアフィン独立であるという．$k+1$ 個のアフィン独立な点からできるアフィン空間を k-**フラット**と呼ぶ．

定義 k-フラット f と m-フラット g は，f の全ての点 p_1 と g の全ての点 p_2 に対して $p_1 - p$ と $p_2 - p$ の内積が 0 になるような点 p で f と g が交わるとき，**直交**するという．

定義 市場サーバ m_i, m_j と，それぞれに割り当てられた部分空間 W_i, W_j に対して，W_i と W_j の各閉包†の積集合がある $N-1$-フラット f に含まれ，いずれの $N-2$-フラットに含まれなく，かつ，f と直交する座標軸の値 a に関して，W_j の閉包の最大値が W_i の閉包の最小値と等しい場合，m_i は m_j に**下流隣接する**と呼ぶ．また，f を W_i と W_j の**接面**と呼ぶ．

市場サーバ m_i は，任意の隣接する市場サーバ m_j に対し，m_i と m_j の接面と直交する座標軸が k 番目のパラメータのものである場合に限り，$m_i.Neighbor[k]$ に m_j のアドレスを格納している．

売注文 x を保持すべき m_i の下流隣接市場サーバの集合を返すアルゴリズムの疑似コード *AddSellOrderSub*，市場サーバ m_i の保持する売注文の集合 $\mathcal{S}_i$ に売注文 x を追加するアルゴリズムの疑似コード *AddSellOrder*，売注文 x を市場サーバ m_i の保持する売注文の集合 $\mathcal{S}_i$ から削除するアルゴリズムの疑似コード *DeleteSellOrder* を以下に示す．

Algorithm *AddSellOrderSub*(int i, Order x, 売り注文の集合 $\mathcal{S}_i$, アドレスの配列 $m_i.Neighbor$)

```
1  /* 入力：市場サーバの番号 i (1 ≤ i ≤ M)，売注文 x，
      m_i の保持する売り注文の集合 S_i，m_i に下流隣接する
      市場サーバのアドレスを格納する配列 m_i.Neighbor．
2  出力：x の複製を保持すべき m_i の下流隣接市場サーバ
      の番号の集合．
3  */
4
5  A := ∅;
6  for j := 1 to N do
7      if α_ij ≤ x.v[j] < β_ij then
8          B := {o | o ∈ S_i, x.v[j] ≤ o.v[j]};
9      else
10         B := {o | o ∈ S_i, β_ij ≤ o.v[j]};
11     endif
12     if B ≠ ∅ then
13         o := B の要素のうち価格が最小である売注文;
14     endif
15     if B = ∅ ∨ (x.price < o.price) then
16         for each n ∈ m_i と下流隣接する市場サーバの
               番号の集合（ただし，アドレス m_i.Neighbor[j]
               を持つ市場サーバは除く） do
17             A := A ∪ {n};
18         next
19     endif
20 next
21 return A;
22
23 end
```

Algorithm *AddSellOrder*(int i, Order x, 売り注文の集合 $\mathcal{S}_i$, アドレスの配列 $m_i.Neighbor$, ベクトル空間 W_i)

```
1  /* 入力：市場サーバの番号 i (1 ≤ i ≤ M)，売注文 x，
      m_i の保持する売り注文の集合 S_i，m_i に下流隣接する市
      場サーバのアドレスを格納する配列 m_i.Neighbor，m_i
      に割り当てられた空間 W_i．
2  出力：S_i に x を追加し，必要なら，隣接する市場サー
      バにも x を保持させる．
3  */
4
5  既に x が S_i に追加されていれば，なにもせずに終了;
6  A := AddSellOrderSub(i, x, S_i, m_i.Neighbor);
7  for each n ∈ A do
8      AddSellOrder(n, x, S_n, m_n.Neighbor, W_n);
9  next
10 S_i := S_i ∪ {x};
```

†集合 S の閉包は，S を含む最小の閉集合

```
11 /* x を S_i に追加することで，S_i のいずれかの要素を，
   保持する必要がなくなった場合を処理 */
12 for each s ∈ S_i (ただし s ≠ x) do
13     if x.v⃗ ∉ W_i ∧ s.v⃗ ≠ W_i then
14         A := AddSellOrderSub(i, s, S_i − {s},
           m_i.Neighbor);
15         if A = ∅ then
16             S_i := S_i − {s};
17         endif
18     endif
19 next
20
21 end
```

Algorithm *DeleteSellOrder*(int i, Order x, 売り注文の集合 $\mathcal{S}_i$, アドレスの配列 $m_i.Neighbor$, ベクトル空間 W_i)

```
1  /* 入力：市場サーバの番号 i (1 ≤ i ≤ M)，売注文 x，
   m_i の保持する売り注文の集合 S_i，m_i に下流隣接する市
   場サーバのアドレスを格納する配列 m_i.Neighbor，m_i
   に割り当てられた空間 W_i.
2  出力：S_i から x を削除し，下流隣接する市場サーバに
   も x を削除させる．
3  */
4
5  m_i が x を保持していなければ，なにもせずに終了;
6  S_i := S_i − {x};
7  /* x を S_i から削除することで，S_i のいずれかの要素
   を，下流隣接市場サーバに保持させる必要が生じる場合
   を処理 */
8  for each s ∈ S_i do
9      A := AddSellOrderSub(i, s, S_i, m_i.Neighbor);
10     for each n ∈ A do
11         AddSellOrder(n, s, S_n, m_n.Neighbor, W_n);

12     next
13 next
14 /* m_i に下流隣接する市場サーバに x を削除させる */
15 for each n ∈ m_i の下流隣接市場サーバの番号の集合
   do
16     DeleteSellOrder(n, x, S_n, m_n.Neighbor, W_n);

17 next
18
19 end
```

4 まとめと今後の課題

4.1 まとめ

本稿では，グリッドコンピューティング環境下で，市場原理に基づき計算機資源を売買する市場において，市場サーバの負荷を分散できるような，市場サーバの分散実行系の提案を行った．

4.2 今後の課題

今後の課題として，本提案方式を実装し，実験による評価を行うことが挙げられる．提案手法の評価項目については，市場サーバ m_i が保持できる最大の注文数 L_i を与えたとき，単位時間当たりに登録される注文の数が増加しても，新たな市場サーバを確保できる限り，各市場サーバの保持する注文数を L_i がそれほど増えないかどうかを評価する予定である．

また，一つの市場サーバで保持する注文の数が大きくなり過ぎる場合に，各市場サーバが担当する部分空間の範囲を，動的に変更するための機構を考案しようと考えている．

今回提案したアルゴリズムでは，登録すべき注文の伝播時間が長いという問題がある．具体的には，今回提案したアルゴリズムでは，ある注文が最初に市場サーバに送られてから，登録が完了するまでに，注文は，最悪の場合，座標軸に沿って並んでいる全てのサーバを順に伝播していくことになる．これを，下流隣接するサーバだけではなく，さらに先にあるサーバに対しても伝播させることで，合計の伝播時間を短縮することができるのではないかと考えている．

また，本稿では，入口サーバが単一である場合を考えたが，将来的に，入口サーバを複数用意し，複数のリクエストを同時に受け付けることができるようにすることを考えている．

参考文献

[1] SETI@Home, http://setiathome.ssl.berkeley.edu/

[2] distributed.net, http://www.distributed.net/

[3] cell computing, http://www.cellcomputing.jp/

[4] Jonathan Bredin, David Kotz and Daniela Rus: Market-based Resource Control for Mobile Agents, Proc. of the 2nd Int'l. Conf. on Autonomous Agents, pp. 197–204, 1998.

[5] Ori Regev and Noam Nisan: The POPCORN Market - an Online Market for Computational Resources, Proc. of the First Int'l. Conf. on Information and Computation Economies (ICE-98), pp. 148–157, 1998.

[6] Rajkumar Buyya, David Abramson, Jonathan Giddy and Heinz Stockinger: Economic Models for Resource Management and Scheduling in Grid Computing, The Journal of Concurrency and Computation: Practice and Experience (CCPE), Vol. 14, pp. 1507–1542, 2002.

[7] Spyros Lalis, Alexandros Karipidis: JaWS: An Open Market-Based Framework for Distributed Computing over the Internet, GRID 2000, pp. 36-46, 2000.