PCグリッド環境での市場原理に基づいた資源共有方式

玉井森彦[†] 柴田直樹[††]
安本慶一[†] 伊藤実[†]

本論文では，PC グリッド環境で市場原理に基づいた資源共有を行うことを目的として，資源の買い手と売り手の注文の登録，適合を処理するマーケットブローカの分散実行方式を提案する．提案方式では，(i) 買い手が必要とする量の資源を複数の売り手から一度に確保することができ，(ii) 各資源の取引価格は市場原理に基づいて自動的に決定される．また，(iii) 複数サーバノードへの負荷の分散により，ユーザ数の増加に対するスケーラビリティが確保できる．シミュレーションにより，提案方式が，集中型のマーケットブローカ方式に比べ，各サーバノードの負荷を大きく減少できること，集中型の場合とほぼ同等の適合結果が実現できることを確認した．

Distributed Market Broker Architecture for PC Grid Resource Sharing

Morihiko Tamai,[†] Naoki Shibata,[††] Keiichi Yasumoto[†]
and Minoru Ito[†]

In order to allow every user to extract aggregated computational power from idle PCs in the Internet, we propose a distributed architecture to achieve a market based resource sharing among users. The advantages of our proposed architecture are as follows: (i) aggregated resources can be bought by one order, (ii) resource prices are decided based on market principles, and (iii) the load is balanced among multiple server nodes to make the architecture scalable w.r.t. the number of users. Through simulations, we have confirmed that the proposed method can mitigate the load at each server node to a great extent, and that our architecture achieves almost the same matching result as the centralized architecture.

1. はじめに

高速ネットワークの普及，PC の高性能化にともない，組織や個人で保有している計算機資源をネットワークを通じて融通し合うグリッドコンピューティングの実現が可能となってきた．ここ数年で，SETI@home [1] や GIMPS [2] などのプロジェクトにより，スーパコンピュータを超える性能を持つ分散コンピューティング環境が実現されてきた．これら PC グリッドシステムは，インターネットに接続された多数の PC の遊休計算機資源を集め，科学技術計算に利用する．これにより，大きな計算性能を，スーパコンピュータ等に比べて非常に安価に実現できる．しかし，多数のユーザを参加させるためには，プロジェクトが行う計算の対象が高い公共性を持つなどの，適切なインセンティブの設定が不可欠である．

上記のような，有志に基づいたユーザ参加型の方式に対し，セルコンピューティング[3] のように，ユーザから集めた遊休計算機資源を，大規模計算を必要とする企業などに有料で提供し，資源提供者には特定の報酬を還元する方式が存在する．しかし，この方式でも，ユーザは資源を提供するだけであり，ユーザ間で自由に資源を融通し合うといった用途には利用できない．

このような問題点を解決するため，これまでに市場原理に基づいた資源共有方式がいくつか提案されている[4)~7)]．これらの方式では，資源を必要とするユーザは，インターネットを介して遊休状態にあるマシンを探し出し，電子通貨を支払うことでその資源を一時的に利用する．また，PC が遊休状態にあるときは，他人に資源を提供することで電子通貨を貯めておくことができる．また，資源の価格は，需要と供給のバランスに基づいて自動的に決定される．

市場原理に基づいた資源共有環境では，資源の買い手は，必要とする資源をできるだけ安い価格で取得でき，資源の売り手は，できるだけ高い価格で資源を提

† 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
†† 滋賀大学経済学部
Faculty of Economics, Shiga University

供できることが望ましい．そのためには資源の売り手と買い手が，互いに適切な取引相手を発見する仕組みが求められる．これを実現するため，既存手法では，**マーケットブローカ**と呼ばれる機能を提供するノードをネットワーク上に配置している．マーケットブローカは，資源の買い手，売り手から注文を受け付け，互いに適合可能な注文の探索，適合結果の通知を行う．

既存手法におけるマーケットブローカは集中制御に基づく処理を行っており，ユーザ数の増加に対するスケーラビリティについては特に考慮されていない．また，注文の適合方法として，単一の買い注文を単一の売り注文と適合させる **1 対 1 適合**のみを扱っている．しかし，資源の買い手が比較的大規模な計算を行う際には，単一の買い注文に対し複数の売り注文を適合させる **1 対多適合**が有用である．

本論文では，マーケットブローカの分散実行方式を提案する．提案方式では，注文の処理に対する負荷を複数サーバノードに分散することで，ユーザ数の増加に対するスケーラビリティを確保する．また，1 対多適合を扱えるようにする．提案方式の基本方針は次のとおりである．まず，注文の集合を資源の量に応じて複数の部分集合に分割する（たとえば，CPU 性能が 1,000 MIPS 未満かそれ以上かで分割）．次に，各注文の部分集合をそれぞれ異なったサーバノードに割り当てる．これにより，各サーバノードが処理すべき注文数を減少させる．一方，各注文がサーバノード間に分散して保持されるため，集中型のマーケットブローカの場合に適合できていた売り注文と買い注文の組が適合できなくなる可能性がある．そこで，複数のサーバノード上で適合する可能性のある注文は，その複製をそれぞれのサーバノードへ保持させる．ただし，サーバノード間の伝送遅延が存在するため，集中型の場合と同様の適合結果を得るためには，複製のためのメッセージを短時間で伝播させる必要があり負荷の増大を招く．必要とされる適合精度と負荷の程度は，提案システムの利用環境や目的によって異なるため，提案システムでは，メッセージの伝播を指定したタイムスロットで行い，タイムスロットの長さを調整することで，システムの利用者が適合精度や負荷の程度を制御できるようにした．

2. 関 連 研 究

市場原理に基づいた計算機資源共有方式に関して，これまでいくつかの研究が行われてきた．

D'Agents[8] では，移動エージェントの移動先ホストの提供ユーザを資源の売り手，エージェントを資源の買い手とし，エージェントの受け入れの可否を売り価格の設定に従って制御する．文献 9)，10) では，仮想通貨の導入により，アドホックネットワーク環境でのパケット転送に対するインセンティブを確立している．これらの方式では，ユーザ間での公平な資源共有を主な目的としている．

PC グリッド環境を対象とした資源共有方式として，文献 4)〜6) では，集中型マーケットブローカの実現方式が提案されているが，マーケットブローカを担当するノードのスケーラビリティや注文の 1 対多適合の方法については考慮されていない．

一方，注文の適合を CAN[11]，Chord[12]，Meghdoot[13] のような P2P 環境下での分散型インデックスサービスを用いて行うことも考えられる．しかし CAN，Chord では，ある値の集合から，特定の値を持つものを探索することはできるが，範囲探索を行うことはできない．一方 Meghdoot は範囲探索をサポートしているが，探索結果の中で最も価格の安い売り注文，または高い買い注文を発見するためには，クライアントが一度探索結果をすべて受信する必要があり，注文数の増加に対するスケーラビリティに欠ける．

3. 市場原理に基づいた資源共有システム

本章では，市場原理に基づいた資源共有システムの概要を述べる．システムは**図 1** に示すように，資源の売り手，買い手，マーケットブローカから構成される．

3.1 構 成 要 素

(a) 資源の売り手　資源の売り手は，計算機資源を買い手に提供することで所定の使用料を得る．売買される計算機資源の種類として，CPU 性能（MIPS），メモリ量（MB），ディスク容量（GB），ネットワーク帯域（Kbps）を考える．これらの各資源の量を資源ベクトル $\boldsymbol{r} = (r_1, r_2, r_3, r_4)$ で表す．売り手は，提供する資源の量 $\boldsymbol{r}$ と，その資源の売り値 $SellPrice$，資源の提供開始時刻 $SellStart$ と終了時刻 $SellEnd$ を指定する．各売り注文 o^s は，資源ベクトル，売り値，提供開始時刻，および，提供終了時刻のこれらの組 $o^s = (\boldsymbol{r}, SellPrice, SellStart, SellEnd)$ によって表される．

(b) 資源の買い手　資源の買い手は，所定の電子通貨を支払うことで，売り手（または，売り手の集合）の計算機資源を利用してタスクを実行する．各タスクは，それを完了するのに必要な計算機資源の量 $\boldsymbol{p}$，実行期間 $ExecDuration$，および，実行期限 $Deadline$ を持つ．

買い手は，タスクの実行に必要な計算機資源の量を売り手の場合と同じく資源ベクトル $\boldsymbol{p}$ で表す．また，

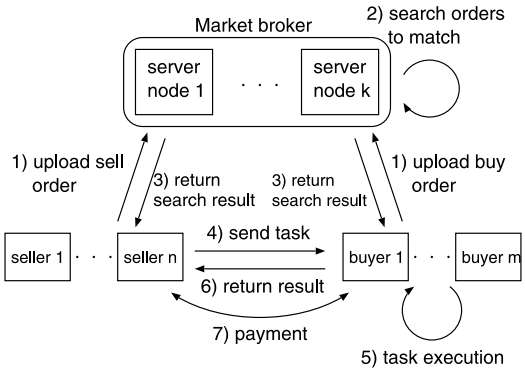

図 1 市場原理に基づいた資源共有システム
Fig. 1 Framework for market based resource sharing.

資源ベクトルで表される計算機資源を利用するための買い値 $BuyPrice$，および，タスクの実行に必要なプログラムとデータを含むコード $Code$ を指定する．各買い注文 o^b は，$o^b = (\boldsymbol{p}, BuyPrice, ExecDuration, Deadline, Code)$ として表される．

(c) マーケットブローカ マーケットブローカは，資源の買い手と売り手から注文を受け取り，3.2 節で述べる注文の適合条件に従って，適合可能な買い注文と売り注文の組を発見する．注文の適合は，東京証券取引所における株式売買などで用いられている方法と同様に行う．すなわち，マーケットブローカが新たな注文を受け取ると，適合条件を満足できる相手注文（買い注文にとっては売り注文，売り注文にとっては買い注文のこと）の集合の中で最適なもの（売り価格が最小の売り注文，または，買い価格が最大の買い注文）を探索する．もし，適合相手が見つかれば，買い手，売り手双方に相手が見つかったことを知らせ，それぞれの注文を自分の保持している注文の集合から削除する．適合相手が見つからなかった場合は，適合相手が見つかるまで注文を保持する．

3.2 適合条件

買い注文 o^b と売り注文 o^s は，以下の条件を満たすとき，**1 対 1 適合**できるとする．

C1 $\forall i \in \{1,2,3,4\}\ (o^b.p_i \leq o^s.r_i)$

C2 $o^b.ExecDuration \leq o^s.SellEnd - o^s.SellStart$

C3 $o^b.Deadline \leq o^s.SellEnd$

C4 $o^s.SellPrice \leq o^b.BuyPrice$

マーケットブローカ m に保持されている売り注文の集合，買い注文の集合をそれぞれ $\mathcal{S}(m)$，$\mathcal{B}(m)$ と表記する．条件 C1–C4 を満足する注文が複数存在する場合，それらの中から最適な注文を選択するため，以下の 2 つの条件を追加する．

C5 マーケットブローカ m が新たな買い注文 o^b を受信した際，$\mathcal{S}(m)$ に属する適合可能な注文の中で，売り価格が最小のものを適合相手とする．

C6 マーケットブローカ m が新たな売り注文 o^s を受信した際，$\mathcal{B}(m)$ に属する適合可能な注文の中で，買い価格が最大のものを適合相手とする．

マーケットブローカが新たな注文を受信し，条件 C1–C4 を満足する注文が存在しなかった場合，それが売り注文である場合は $\mathcal{S}(m)$ に，買い注文である場合は $\mathcal{B}(m)$ に追加される．また，$o^s.SellEnd$ または $o^b.Deadline$ を経過した売り注文 o^s もしくは買い注文 o^b は，マーケットブローカの保持する注文の集合から取り除かれ，注文を登録したユーザへ適合が失敗したことが伝えられる．

以上の適合条件は文献 4)，6) でも同様に用いられている．

1 対多の適合条件 買い手がタスクの実行に比較的多くの計算資源を必要とする場合，1 つのタスクを複数のサブタスクに分割し，それぞれを異なった売り注文に一度に適合させることができれば有用である．以下では，このような 1 対多適合を扱えるよう，1 対 1 の適合条件の自然な拡張を行う．

O^b を n 個（ただし，$n \geq 2$）のサブタスクを含む買い注文とする．O^b を次のように定義する：$O^b.p$=$\{(o^b_1.p_1, o^b_1.p_2, \ldots, o^b_1.p_4), \ldots, (o^b_n.p_1, o^b_n.p_2, \ldots, o^b_n.p_4)\}$，$O^b.ExecDuration$，$O^b.Deadline$, $O^b.BuyPrice$, $O^b.Code = \{code_1, \ldots, code_n\}$．ただし，$i$ 番目のサブタスクは，プログラムおよびデータ $code_i$ の実行のために資源 $(o^b_i.p_1, o^b_i.p_2, o^b_i.p_3, o^b_i.p_4)$ を必要とし，すべてのサブタスクの実行時間および実行期限はそれぞれ $O^b.ExecDuration$ および $O^b.Deadline$ を超えないものとする．また，すべてのサブタスクを実行するために使用できる予算を $O^b.BuyPrice$ とする．

買い注文 O^b に対し，以下の条件を満足する売り注文の集合 $S = \{o^s_1, \ldots, o^s_n\}$ が存在するとき，O^b と S は **1 対多適合**するものとする．

D1 $\forall i \in \{1,2,\ldots,n\}, \forall j \in \{1,2,\ldots,4\}\ (o^b_i.p_j \leq o^s_i.r_j)$

D2 $\forall i \in \{1,2,\ldots,n\}\ O^b.ExecDuration \leq o^s_i.SellEnd - o^s_i.SellStart$

D3 $\forall i \in \{1,2,\ldots,n\}\ O^b.Deadline \leq o^s_i.SellEnd$

D4 $O^b.BuyPrice \geq \sum_{i=1}^{n} o^s_i.SellPrice$

条件 D10–D4 を満足する売り注文の集合 S が存在しない場合は，D1–D4 を満足する売り注文の集合が現れるまで O^b はマーケットブローカに保持される．また，$O^b.Deadline$ を経過した買い注文 O^b は，マー

ケットブローカの保持する注文の集合から取り除かれ，注文を登録したユーザへ適合が失敗したことが伝えられる．

3.3 分散化の基本方針

提案方式では，以下のような方針でマーケットブローカの分散化を実現する．

(1) マーケットブローカに登録される注文の全体集合を複数に分割し，各部分集合をそれぞれ異なったノードに管理させる（以下では，注文の管理ノードを**サーバノード**とよぶ）．
(2) 売り手の中からサーバノードを自律的に選択する．
(3) 特定のサーバノードの負荷が増加した際には，新たなサーバノードを確保し，負荷を分散する．

上記 (2) を実現するため，資源の売り手，買い手は，サーバノードを使用する際に，使用料として所定量の電子通貨を支払うものとする．また，上記 (3) を実現するため，特定のサーバノードの負荷がある閾値を超えた場合には，そのサーバノードが管理する注文の集合を複数に分割し，新たに確保したサーバノードに一部を管理させる．以下では，主に上記 (1) を実現するための方法に焦点をあてる．

4. マーケットブローカの分散実行方式

本章では，初めに 1 対 1 適合の際のアルゴリズムについて述べる．次に，4.6 節で 1 対多適合を実現するための拡張方法について述べる．

4.1 注文の表記

買い注文 o^b，売り注文 o^s に対し，$\forall i \in \{1,\ldots,4\}$ $o^b.v_i = o^b.p_i$，$o^s.v_i = o^s.r_i$ とする．また，$o^b.v_5 = o^b.ExecDuration$，$o^s.v_5 = o^s.SellEnd - o^s.SellStart$，$o^b.v_6 = o^b.Deadline$，$o^s.v_6 = o^s.SellEnd$ とする．$o^b.v_i$ の i の範囲をより一般的に $\{1,\ldots,d\}$ とし，ベクトル $(o^b.v_1,\ldots,o^b.v_d)$，$(o^s.v_1,\ldots,o^s.v_d)$ をそれぞれ $o^b.\boldsymbol{v}$，$o^s.\boldsymbol{v}$ と表記する．

各 i に対し，$o^b.v_i$，$o^s.v_i$ が取り得る値の範囲はあらかじめ決まっているものとし，それを $[min_i : max_i]$ と表記する．d 次元空間 $W = [min_1 : max_1] \times \ldots \times [min_d : max_d]$ を考えると，$o.\boldsymbol{v} \in W$ より，注文 o を W 内の座標として表すことができる．W を**全体領域**とよぶ．以下簡単のため，一般性を失うことなく，$d = 2$ の場合について説明する．

注文の適合方法は売り買いで対称であるため，以下では，すでに保持されている売り注文に対し，新たな買い注文を適合させる場合について述べる．

4.2 全体領域の分割

M をサーバノード数とし，サーバノードの集合を $\{m_1, m_2, \ldots, m_M\}$ とする．提案方式では，全体領域 W を M 個の部分領域に分割し，各部分領域を異なったサーバノードに割り当てる．以下では，全体領域に対し注文が一様に分布する場合を想定し，各部分領域の大きさを同一に設定する．また，サーバノード m_i の担当領域を R_i と表記する（**図 2** (a)）．

ユーザが新たな注文 o をマーケットブローカへ登録するとき，あるサーバノード m_s を選び，m_s へ注文 o を送信する☆．次に，m_s から，$o.\boldsymbol{v} \in R_t$ となるサーバノード m_t まで注文 o の転送を行う．この転送には，CAN[11] で述べられている方法を用いる．m_s から m_t への転送にかかる平均ホップ数は $O(dM^{1/d})$ となる[11]．

サーバノード m_t が $o^b.\boldsymbol{v} \in R_t$ となる買い注文 o^b を受信したとき，m_t はまず，3.2 節で述べた条件 C1–C4 を満足する売り注文を探索する．もしそのような売り注文が見つからなかった場合は，3 章で述べたように，o^b を待ち状態にある買い注文の集合 $\mathcal{B}(m_t)$ に追加する．

注文が M 個のサーバノードに一様に受信される場合，以上のように全体領域の分割を行えば，集中型のマーケットブローカに比べ，各サーバノードの負荷を $1/M$ に減少させることができることが期待できる．一方，このままでは，もし買い注文 o^b と売り注文 o^s が互いに異なったサーバノードに受信された場合，o^b と o^s は適合できない．たとえば，$o^b.\boldsymbol{v} \in R_7$，$o^s.\boldsymbol{v} \in R_6$ の場合，o^b，o^s はそれぞれサーバノード m_7，m_6 によって受信される．したがって，もし o^b，o^s が適合条件 C1–C4 を満足する場合でも，適合することができない．以下では，この問題を注文の複製により解決する．

4.3 注文の複製

以下では，提案方式の動作を具体的な例を示しながら説明する．

売り注文 o^s の**有効領域** $R(o^s)$ を $R(o^s) = [min_1 : o^s.v_1] \times \ldots \times [min_d : o^s.v_d]$ と定義する．$R(o^s)$ は，$o^b \in R(o^s)$ となる買い注文 o^b が新たに送信された際，3.2 節の条件 C4 を満足すれば，o^s が o^b と適合可能であることを意味している．

まず，サーバノード m_7 に売り注文 o^s_1 のみが登録されているとする．次に，新たな売り注文 o^s_2 がサーバノード m_6 に登録されたとする．o^s_1，o^s_2 の有効領域はそれぞれ図 2 (b) のようになる（図 2 (b) から図 2 (d)

☆ 各ユーザは，前もって少なくとも 1 つのサーバノードのアドレスを知っているものと仮定する．

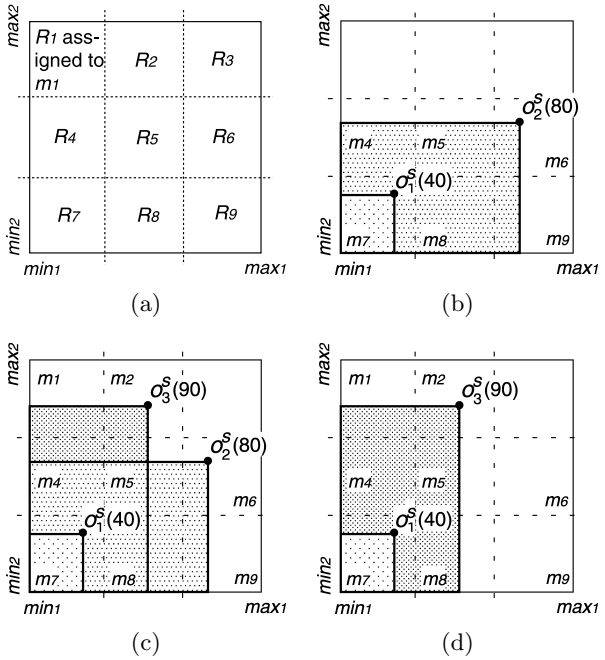

図 2 部分領域と各サーバノードへの注文の登録
Fig. 2 Subregion and registered orders.

において，( ) 内の数字は，その注文の売り価格を表している）．図 2 (b) より，o_2^s の有効領域はサーバノードの集合 $\{m_4, m_5, m_7, m_8, m_9\}$ に及んでいる．そこで，o_2^s の複製をこれらのサーバノードに保持させる．

次に，新たな売り注文 o_3^s がサーバノード m_2 に到着したとする．o_3^s の有効範囲は，図 2 (c) のようにサーバノードの集合 $\{m_1, m_4, m_5, m_7, m_8\}$ に及ぶが，この場合，m_7，m_8 に o_3^s の複製を保持させる必要はない．なぜなら，もし m_7 または m_8 に買い価格が 80 以上の買い注文が到着した場合，$o_2^s.SellPrice = 80 \leq o_3^s.SellPrice = 90$ より，その買い注文は o_2^s と優先的に適合するためである．したがって，o_3^s の複製はサーバノードの集合 $\{m_1, m_4, m_5\}$ にのみ保持させる．この例のように，ある売り注文 o_a^s の有効領域が，それよりも価格の安い売り注文 o_b^s の有効領域と重なる場合，o_a^s の実質的な有効領域は $R(o_a^s) - R(o_b^s)$ となる．これを，o_a^s の**実有効領域**とよび，$RA(o_a^s)$ と表記する．

4.4 注文の適合と削除

次に，新たな買い注文 o_1^b がサーバノード m_6 に到着し，o_2^s と適合したとする．このとき，適合処理（買い手，売り手の間で資源の受け渡しのために必要な情報の交換など）を終えた後，m_6 から o_2^s を削除する．また，o_2^s の複製をサーバノードの集合 $\{m_4, m_5, m_7, m_8, m_9\}$ から削除する（図 2 (d)）．o_2^s を削除したあと，o_3^s の実有効領域がサーバノード m_7，m_8 に及ぶため，m_7，m_8 に注文 o_3^s の複製を保持させる．

ここで，新たな買い注文 o_2^b がサーバノード m_8 に送信され，かつ，o_1^s と o_1^b の適合による o_2^s の削除処理が進行中であるとする．このとき，o_2^s はすでに o_1^b と適合した後であるにもかかわらず，サーバノード m_8 において o_2^s と適合する可能性がある．このような際の一貫性を維持するため，サーバノードは，注文の複製との適合が起こった際に，注文のオリジナルを保持するサーバへ，その注文が適合前であるかどうかを問い合わせるものとする．そのため，各注文の複製には，そのオリジナルを保持するサーバノードのアドレスを保持させる．

4.5 注文の複製と削除のためのメッセージ

(a) 複製メッセージ サーバノード m_i の**下流隣接サーバノード**を，W 上で座標 $(min_1, \ldots, min_d)$ に向かって m_i と隣接する領域の担当サーバノードと定義する．たとえば図 2 (a) において，m_5 の下流隣接サーバノードの集合は $\{m_4, m_8\}$ である．サーバノード m_i が新たな売り注文 o^s を受信した際，m_i の下流隣接サーバノード m_j に対し $RA(o^s) \cap R_j \neq \emptyset$ が成り立つとき，m_j に o^s の複製を保持させるためのメッセージを送信する．$RA(o^s) \cap R_j \neq \emptyset$ が成り立つかどうかは，m_i に保持されている売り注文の集合から，m_i 自身で計算できる．複製メッセージを受信し

たサーバノードも同様に，必要であれば下流隣接サーバノードへ複製メッセージを送信する．

たとえば，サーバノード m_6 で新たな売り注文 o_2^s が受信されたとき，o_2^s の複製メッセージは次のように送信される．まず，サーバノード m_6 が複製メッセージをサーバノード m_5，m_9 に送信する．次に，m_5，m_9 からサーバノード m_4，m_8 に複製メッセージが送信され，最後に，サーバノード m_7 で複製メッセージが受信される．

(b) 削除メッセージ 買い注文 o^b が，サーバノード m_i 上で o^s の複製と適合したとする．4.4 節で説明したように，ある注文の複製との適合が発生した際には，まず，その注文のオリジナルがまだ存在するかどうかを確認する．o^s のオリジナルが存在する場合，o^b はサーバノード m_i において o^s との適合が確定し，結果として，o^s のオリジナルと m_i に保持されている o^s の複製がそれぞれ削除される．

次に，o^s の複製が他のサーバノードに保持され続けることを防ぐため，o^s のオリジナルを保持していたサーバノード（m_j とする）から，m_j の下流隣接サーバノードに対して o^s の削除メッセージを送信する．削除メッセージを受信した各サーバノードも同様に削除メッセージを下流隣接サーバノードへ送信する．

(c) タイムスロットの導入 上記の方法では，短時間に多くの注文が 1 つのサーバノードに送信された場合，そのサーバノード，および下流隣接サーバノードの負荷が，複製，削除メッセージの送信処理によって著しく高まることが予想される．そこで，ある固定時間のタイムスロット T を導入し，T の間に新たに登録された注文，受信された複製，削除メッセージの結果をマージすることで効率化する．各サーバノードは，タイムスロット終了時に，各メッセージのマージの結果，新たに必要となった複製，削除メッセージのみを隣接サーバノードへ送信する．

タイムスロットを使用することで，複製メッセージの送信に遅延が発生するため，集中型のマーケットブローカを使用する場合と比べ，適合結果に差が生じることが予想される．しかし，一般的に注文の買い価格，売り価格は，資源の量が増えるに従って高く設定されるため，多くの注文は，そのオリジナルを保持するサーバノード，および，それに隣接したサーバノード上で適合すると考えられ，その影響は少ないと思われる．

4.6 1 対多適合

n 個のサブタスクからなる買い注文を O^b とする．$\{o_1^b, \ldots, o_n^b\}$ をサブタスクに対する買い注文の集合とする．各 o_i^b を**副買い注文**とよぶ．3.2 節で定義したように，1 対多適合の際には，各副買い注文の買い価格は考えず，使用可能な予算 $O^b.BuyPrice$ のみが与えられる．買い手は，n 個の副買い注文をそれぞれ独立した注文としてマーケットブローカに登録する．その際，各副買い注文の $BuyPrice$ は未定義値 $undefined$ に設定しておく．提案方式では，以下のような，2 相コミットメントに基づいた方法により 1 対多適合を実現する．

サーバノード m_i が新たな注文 o_i^b を受信し，かつ，$o_i^b.BuyPrice = undefined$ である場合，m_i は 1 対 1 適合の条件 C1–C3 を満足する売り注文の中で最小の価格を持つ売り注文 o_i^s を見つけ，o_i^b と**仮適合**させる．また，$o_i^s.SellPrice$ の値とともに，o_i^b が仮適合したことを買い手に伝える．o_i^s が見つからなかった場合は，条件 C1–C3 を満足する売り注文が登録されるまで待つ．仮適合中の注文は，他の注文と仮適合，または適合することが可能である．

買い手が送信したすべての副買い注文が仮適合し，かつ，$O^b.BuyPrice \geq \sum_{i=1}^{n} o_i^s.SellPrice$ が満たされる場合，買い手は，仮適合の発生したすべてのサーバノードへ**確認メッセージ**を送信する．一方予算が不足する場合は，予算を増加させるよう買い手に注意を促す．

確認メッセージを受信したサーバノードは，1 対 1 適合の際と同様に，その注文のオリジナルが存在するか確認する．オリジナルが存在する場合は，それを適合確定もキャンセルも可能な**中間状態**に移行させ，買い手に **ack メッセージ**を送信する．オリジナルが存在しない場合は，買い手に **nack メッセージ**を送信し，再度，副買い注文と仮適合できる売り注文を探索する．中間状態に移行した注文は，他の注文と仮適合，または適合することはできない．

買い手がすべてのサーバノードから ack メッセージを受信した場合，O^b の適合が確定し，すべてのサーバノードへ**確定メッセージ**を送信する．確定メッセージを受信したサーバノードは，1 対 1 適合の場合と同様に注文のオリジナルおよび複製の削除を行う．一方，少なくとも 1 つのサーバノードから nack メッセージを受信した場合，ack メッセージを受信したすべてのサーバノードへ**キャンセルメッセージ**を送信し，中間状態を解除する．

上記のアルゴリズムにおいて，複数の買い手がそれぞれ複数の副買い注文を市場サーバへ登録し，仮適合が発生したとする．このとき，異なった買い手の副買い注文が同じ売り注文に適合していた場合，次のような競

合が発生する可能性がある．たとえば，ユーザ 1，ユーザ 2 の買い注文がそれぞれ $\{o_1^s, o_3^s, o_4^s\}$，$\{o_2^s, o_3^s, o_4^s\}$ と仮適合したとする．次に，両ユーザがほぼ同時刻に確認メッセージを送信開始し，o_3^s を保持するサーバノード上ではユーザ 1 のメッセージが，o_4^s を保持するサーバノード上ではユーザ 2 のメッセージが，それぞれ先に受信されたとする．この場合，両ユーザとも買い注文の適合に失敗する．

このような競合状態を避けるため，確認メッセージを送信する際には，より小さい IP アドレスを持つサーバノードへ先に送信し，かつ，次のサーバノードへのメッセージの送信は，現在送信中のサーバノードから ack メッセージが受信された後に行うようにする．

5. 実験結果

提案方式による負荷分散の効果を調べるため，シミュレーションによる実験を行った．実験では，サーバノードの数を 1 から 729 まで変化させ，サーバノードあたりに処理されるメッセージ数の平均値を測定した．メッセージは，注文の登録メッセージ，複製メッセージ，削除メッセージの 3 種類である．30,000 個の注文を生成し，売り注文か買い注文かはそれぞれ 1/2 の確率で決定した．資源の種類は 2 種類および 3 種類とし，各資源の量は区間 [0 : 100] の一様乱数に従い決定した．各サーバノードの担当領域は同じ大きさとし，サーバノード間の通信遅延は区間 [10 : 160] の一様乱数に従い決定した（単位は ms）．注文の買い価格，売り価格は，資源 i の量を v_i としたとき，平均 $\Sigma_i v_i$，標準偏差 30 の正規乱数により決定した．注文の登録メッセージは，平均 10 ms のポアソン到着に従い発生させ，タイムスロットは 1,000 ms に設定した．

実験結果を**図 3** に示す．図には，サーバノードあたりに処理されたメッセージ数の平均値とともに，最大値，最小値も示した．なお，本論文ではノード数が 1 から 729，注文数が 30,000 の場合の実験結果を示したが，これより大規模な注文数を扱う場合でも，適切なノード数を設定することで負荷分散の傾向としては図 3 と同様の結果が得られる．図 3 はその傾向を示すための一例として，具体的にノード数 1 から 729，注文数 30,000 という設定のもとでの実験結果を示したものである．

図 3 より，資源の種類が 2 種類の場合，3 種類の場合のいずれにおいても，サーバノード数の増加にともない，サーバノードあたりの平均処理メッセージ数を大きく減少させることができることが分かる．特に資源の種類が 2 種類である場合に着目すると，集中型の

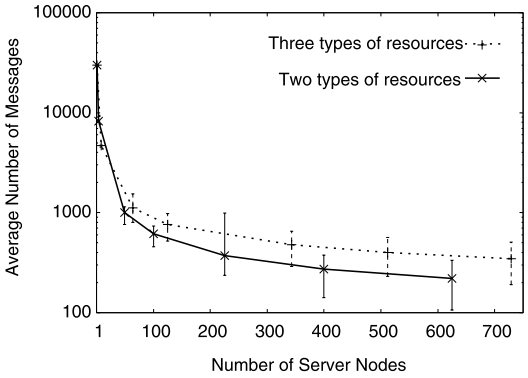

図 3 サーバノードあたりに処理される平均メッセージ数
Fig. 3 Average number of messages received by each server node.

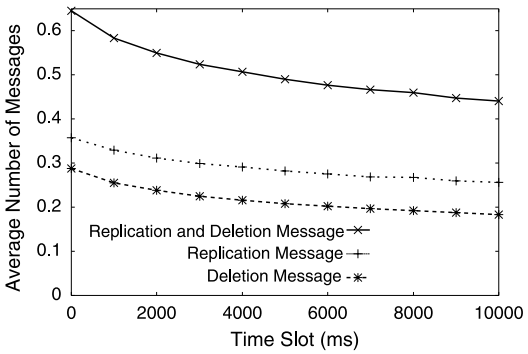

図 4 タイムスロットとサーバノードあたりに処理される平均メッセージ数の関係
Fig. 4 Time slot vs. average number of messages received by each server node.

マーケットブローカの場合に処理されるメッセージ数が 30,000 であるのに対し，サーバノード数を 625 まで増加させることで，サーバノードあたりの平均処理メッセージ数を 219 まで減少させることができることが分かる．また，このときのサーバノードあたりの処理メッセージ数の最大値は 334 であることから，処理メッセージ数の最大値に関して見た場合でも，集中型の場合と比べ処理メッセージ数を大きく削減できることが分かる．

次に，タイムスロットの導入によるメッセージ数削減の効果を調べるため，タイムスロットを 0 から 10,000 ms まで変化させたときの，サーバノード 1 台が 1 秒あたりに受信した複製メッセージ数と削除メッセージ数の合計の平均値を求めた．資源の種類は 2 種類であり，サーバノード数は 625 である．他のパラメータ値は前の実験と同一である．実験結果を**図 4** に示す．

図 4 より，タイムスロットを 10 秒まで増加させる

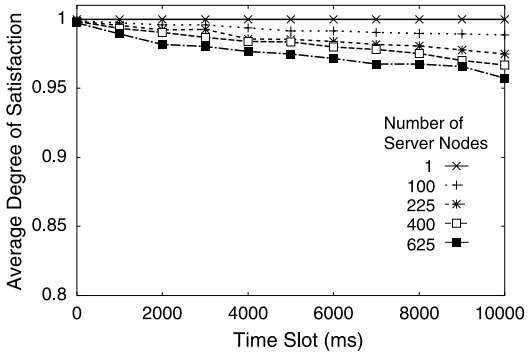

図 5 平均満足度
Fig. 5 Average degree of satisfaction.

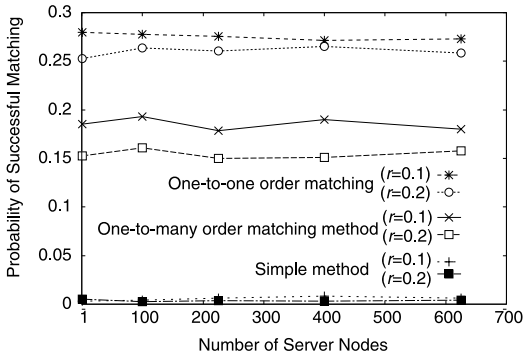

図 6 1 対多適合の効果
Fig. 6 Effectiveness of one-to-many matching.

ことで，1 秒あたりに受信する複製，削除メッセージ数を 0.65 から 0.45 程度まで減少させることができることが分かる．

次に，集中型のマーケットブローカに対し，提案方式による適合結果がどの程度異なるかを調べた．実験では，注文の適合結果に対するユーザの満足度 S を以下のように定義し，S の平均値が集中型の場合に比べどの程度異なるかを測定した．

新たに送信された注文 o_1 が，すでにマーケットブローカに保持されていた注文 o_2 と適合したとする．o_1 が買い注文の場合，$D = o_1.BuyPrice - o_2.SellPrice$，$o_1$ が売り注文の場合，$D = o_2.BuyPrice - o_1.SellPrice$ と定義する．集中型のマーケットブローカの場合，o_1 に対し，D の値が最大となる注文 o_2 がつねに選択される．一方提案方式では，サーバノード間のメッセージの送受信に遅延が存在するため，D の値が最大となる注文 o_2 が，つねに選択されるとは限らない．サーバノード上での各適合において，提案方式を使用した際の適合結果による D の値を D_L，また，集中型のマーケットブローカを使用した際の適合結果による D の値を D_G とし，ユーザの適合結果に対する満足度 S を $S = D_L/D_G$ と定義する．ここで，$0 \leq D_L \leq D_G$ である．また，$D_G = 0$ のとき，$S = 1$ とする．

注文を 30,000 個発生させた際の，全適合における S の平均値を測定した．資源の種類は 2 種類であり．サーバノード数を 1 から 625，タイムスロットを 0 から 10,000 ms まで変化させた．その他のパラメータ値ははじめの実験と同一である．測定結果を**図 5** に示す．

図 5 より，いずれのサーバノード数においても，集中型のマーケットブローカに匹敵する満足度（0.95 以上）が達成できることが分かる．

次に，1 対多適合の有効性を検証するため，1 対多適合を以下で述べる単純な方法，および，4.6 節で述べた方法を用いて行った場合の，適合の成功確率の比較を行った．単純な方法では，買い価格が設定されている n 個の買い注文を副買い注文（買い価格は，はじめの実験と同じ方法で決定）と見なし，それぞれを独立した買い注文としてマーケットブローカに登録する．このとき，n 個の買い注文がすべて適合すれば，1 対多適合が成功したとする．一方提案方式では，単純な方法を用いる際に発生させたものと同一の買い注文に対し，買い価格の和を予算とし，それぞれの買い価格を $undefined$ に設定したものを副買い注文としてマーケットブローカに登録する．すべての副買い注文が仮適合し，かつ，予算が売り価格の合計未満であれば，適合失敗とする．また，予算が売り価格の合計以上である場合でも，少なくとも 1 つのマーケットブローカから nack メッセージが返信された場合，適合失敗とする．

実験では，1 対多適合を行う買い注文数として全買い注文数の 1 割から 2 割程度を想定し，売り注文数を 15,000，1 対 1 適合および 1 対多適合を行う買い注文数をそれぞれ 13,500 と 1,500，もしくは 12,000 と 3,000 に設定した．1 対多適合を行う買い注文を構成する副買い注文の個数は，平均値が 10 の指数分布に従い決定した．タイムスロットは 1,000 ms に設定した．資源の種類は 2 種類であり，サーバノード数を 1 から 625 まで変化させた．その他のパラメータ値ははじめの実験と同一である．実験結果を**図 6** に示す．なお図中の r は，全買い注文数に対する，1 対多適合を行う買い注文数の割合を示している．また参考として，1 対 1 適合の適合率も図示している．

図 6 より，単純な方法での成功率はほぼ 0 に近いことが分かる．また，提案方式による 1 対多適合の成功率は 0.16 程度であり，単純な方法に比べ高い成功率

を達成できることが分かる．

単純な方法での適合成功率が低い原因として，単純な方法では副買い注文がそれぞれ独立に売り注文と適合する必要があるため，1 対 1 適合の成功率を p，副買い注文数を n としたとき，適合成功率が約 p^n となるためだと考えられる．一方提案方式では，売り注文の売り価格の合計が，予算（副買い注文の買い価格の合計）を下回れば適合可能であるため，単純な方法に比べ高い適合率を達成できたと考えられる．

1 対多適合の評価項目として，上で述べたような，(i) 1 対多適合の成功率がどれだけ高いか，に加え，(ii) 適合した売り注文の売り価格の和がどれだけ小さいか，があげられる．(ii) に関しては，各副買い注文に対する売り注文の仮適合が，その時点で最も安い売り注文に対して行われるため，タイムスロットが 0 で，かつメッセージ伝送遅延のない場合においては，1 対多適合の適合相手として可能な売り注文の集合のうち，最も売り価格の和が小さくなるものが選択される．(i) に関しても，タイムスロットが 0 で，かつメッセージ伝送遅延のない場合においては，その時点で最も安い売り注文の集合が適合し，かつ仮適合中の売り注文が自分以外の買い注文と適合することもないため，その時点で最も高い適合率が得られる．

6. おわりに

本論文では，市場原理に基づいた資源共有環境における，注文の 1 対 1 および 1 対多適合が行えるマーケットブローカの分散実行方式を提案した．

提案方式のデメリットとして，サーバノード間での通信によるシステム全体の応答時間の悪化や，注文の複製によるストレージ資源の総使用量の増加があげられる．しかし分散化には，サーバノードあたりが処理すべきメッセージ数の削減に加え，各サーバノードが保持すべき注文数を少なく抑えられるという利点もある．これには，注文の適合相手の探索に必要な処理時間を集中型に比べ減少させることができるという効果がある．またストレージ資源のコストが，単一で大規模なものを用意するより，小規模なものを複数用意する方が安価に低く抑えられるような場合には，本方式を使用することでコストの削減が期待できる．したがって，上で述べたようなデメリットを考慮しても，サーバノードの分散実装方式の実現は重要であると考える．

本論文では，注文の分布に関して最も単純な場合である一様分布を想定し，各サーバノードの担当領域を決定した．しかし，特定のサーバノードへ負荷が集中するような分布の場合は，サーバノードの負荷に応じて担当領域をさらに分割し，複数サーバノードへ負荷を分散することが求められる．このような動的な領域分割方式の考案については今後の課題である．

また今後，提案手法を実際に複数の計算機上で分散実行し，ユーザの要求に対する応答時間等のより詳細な性能評価を行う予定である．

参考文献

1) SETI@home. http://setiathome.ssl.berkeley.edu/
2) Mersenne Prime Search. http://www.mersenne.org/prime.htm
3) cell computing. http://www.cellcomputing.jp/
4) Regev, O. and Nisan, N.: The POPCORN Market — an Online Market for Computational Resources, *Proc. 1st Int'l Conf. on Information and Computation Economies* (*ICE-98*), pp.148–157 (1998).
5) Amir, Y., Awerbuch, B. and Borgstrom, R.S.: A Cost-Benefit Framework for Online Management of a Metacomputing System, *Proc. 1st Int'l Conf. on Information and Computation Economies* (*ICE-98*), pp.140–147 (1998).
6) Lalis, S. and Karipidis, A.: JaWS: An Open Market-Based Framework for Distributed Computing over the Internet, *Proc. GRID 2000*, pp.36–46 (2000).
7) Abramson, D., Buyya, R. and Giddy, J.: A computational economy for grid computing and its implementation in the Nimrod-G resource broker, *Future Generation Computer Systems* (*FGCS*) *Journal*, Vol.18, No.8, pp.1061–1074 (2002).
8) Bredin, J., Kotz D. and Rus, D.: Market-based Resource Control for Mobile Agents, *Proc. 2nd Int'l Conf. on Autonomous Agents*, pp.197–204 (1998).
9) Buttyán, L. and Hubaux, J.: Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks, *ACM/Kluwer Mobile Networks and Applications* (*MONET*), Vol.8, No.5, pp.579–592 (2003).
10) Chen, K. and Nahrstedt, K.: iPass: An Incentive Compatible Auction Scheme to Enable Packet Forwarding Service in MANET, *Proc. 24th Int'l Conf. on Distributed Computing Systems* (*ICDCS 2004*), pp.534–542 (2004).
11) Ratnasamy, S., Francis, P., Handley, M., Karp, R. and Shenker, S.: A Scalable Content-Addressable Network, *Proc. ACM SIGCOMM*

2001, pp.161–172 (2001).
12) Stoica, I., Morris, R., Karger, D., Kaashoek, M.F. and Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, *Proc. ACM SIGCOMM 2001*, pp.149–160 (2001).
13) Gupta, A., Sahin, O.D., Agrawal, D., and Abbadi, A.E.: Meghdoot: Content-Based Publish/Subscribe over P2P Networks, *Proc. Middleware 2004*, pp.254–273 (2004).

(平成 17 年 5 月 10 日受付)
(平成 17 年 11 月 1 日採録)

玉井 森彦（学生会員）
2002 年岡山県立大学情報工学部情報システム工学科卒業．2004 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．現在，同大学院博士後期課程在学中．マルチメディア通信システム，分散処理方式に興味を持つ．

柴田 直樹（正会員）
2001 年大阪大学大学院基礎工学研究科情報数理系専攻博士後期課程修了．2001 年より奈良先端科学技術大学院大学情報科学研究科助手，2004 年より滋賀大学経済学部助教授．分散システム，マルチメディア通信システム，ITS，遺伝的アルゴリズム等の研究に従事．

安本 慶一（正会員）
1991 年大阪大学基礎工学部情報工学科卒業．1995 年同大学大学院博士後期課程退学後，滋賀大学経済学部助手．2002 年より奈良先端科学技術大学院大学情報科学研究科助教授．博士（工学）．分散システムに関する研究に従事．IEEE/CS，ACM 各会員．

伊藤 実（正会員）
1977 年，1979 年，1983 年にそれぞれ大阪大学基礎工学部卒業，基礎工学研究科博士前期課程修了，基礎工学研究科博士後期課程修了．1979 年より大阪大学基礎工学部助手．1986 年より大阪大学基礎工学部講師．1989 年より大阪大学基礎工学部助教授．1993 年 4 月より現在，奈良先端科学技術大学院大学情報科学研究科教授．関係データベース理論，オブジェクト指向データベースのアプリケーション，DNA プローブ等の研究に従事．ACM，IEEE 各会員．