DEWS2007 D1-3

資源管理機構を持つP2Pグリッド実行基盤の検討と試作

水江　真登† 　廣川　雅基† 　加藤　宏章†† 　小林　孝史††† 　上島　紳一†††

†,††† 関西大学大学院総合情報学研究科　〒 569–1095 大阪府高槻市霊仙寺町 2-1-1
†† 関西大学総合情報学部　〒 569–1095 大阪府高槻市霊仙寺町 2-1-1
E-mail: †,†††{fb5m130,fb5m129,fa30095}@edu.kansai-u.ac.jp, ††††{kobayasi,ueshima}@res.kutc.kansai-u.ac.jp

あらまし　近年, 多数の計算機を統合的に扱うグリッドコンピューティング技術や個々の計算機が自律的にネットワークを構成する P2P 技術が注目を集めている．本稿では P2P 環境における PC グリッド実現のための P2P グリッド実行基盤を提案する．提案実行基盤では参加 PC(ピア) を計算機資源として扱い，分散的にそれらの計算機資源を管理する．そのため，P2P 方式による資源管理機構を構成する．実行基盤の利用ピアは自身の資源情報を管理機構に登録し, 共有資源とする．実行基盤を利用して構成する計算グリッドや分散ストレージを本稿ではサービスと呼ぶ．提案管理機構はサービスの稼働状況に従って，登録情報を元にサービスにピアを割当てる．本手法の有効性を確認するため，資源管理機構のシミュレーション，計算グリッドサービスの試作を行う．

キーワード　グリッドコンピューティング，P2P，オーバーレイネットワーク，並列・分散 DB

Prototyping and Evaluation of P2P Grid Platform with Resource Management Mechanism

Masato MIZUE†, Masaki HIROKAWA†, Hiroaki KATO††, Takashi KOBAYASHI†††, and Shinichi UESHIMA†††

† †††Graduate School of Informatics, Kansai University Ryozenji 2-1-1, Takatsuki-shi, Osaka, 569–1095 Japan
†† Faculty of Informatics, Kansai University　Ryozenji 2–1–1, Takatsuki-shi, Osaka, 569–1095 Japan
E-mail: †,†††{fb5m130,fb5m129,fa30095}@edu.kansai-u.ac.jp, ††††{kobayasi,ueshima}@res.kutc.kansai-u.ac.jp

Abstract　Researches on Grid computing which several computers work concurrently, and researches on P2P systems which each peer generates network autonomously are gaining focuses recently. This paper proposes a PC grid platform in P2P settings. In this platform, a PC(peer) is considered as a storage or computational resource, and its resource is managed distributively in a P2P manner by the resource management mechanism proposed in this paper. Peers on a platform register their resource information to this mechanism, and share their resources. We consider Processing Grid and Distributed Storage as services in this platform. When the services request the resources, the resource management mechanism assigns the services to peers according to the registered information. The authors have constructed a prototype, and have simulated to view the effects on resource management mechanism.

Key words　GridComputing, P2P, OverlayNetwork, Parallel · Distributed DB

1. はじめに

近年，ネットワーク上に分散した様々な計算機資源を仮想的に統合して利用するグリッドコンピューティング技術が注目を集めている [1] [2]．特に PC によって構成されるグリッドは PC グリッドと呼ばれ，多数の PC の CPU 空き時間を有効活用する PC グリッドプロジェクトが進められている [3] [4]．しかし，これらの PC グリッドプロジェクトは PC の余剰資源を特定組織や企業が利用する方式として設計されており，PC の計算機資源を共有し，個人が利用する技術の研究はまだ少ない [5].

本稿では，P2P 環境において PC グリッドを実現することを目的とし，資源管理機構を備えた P2P グリッド実行基盤を提案する．本実行基盤を利用する PC(ピア) は自律的な処理により P2P オーバーレイネットワークを生成し，計算グリッドによる分散処理や分散ストレージを実現する．本稿ではこれらの計算グリッドや分散ストレージをサービスと呼ぶ．実行基盤利用者はサービスを自由に作成し，任意に利用可能となる．資源管理機構は P2P 技術で用いられる空間管理手法を応用して，CPU

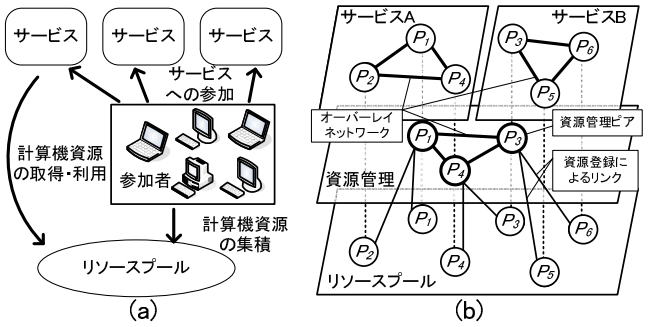

図 1　(a) 提案実行基盤上の PC グリッド，(b) ネットワークの多重構造

Fig. 1　(a)System concept, (b)Overlay network structure

性能と HD 容量を座標軸とした論理空間を BATON [6] 上に構成し，参加ピアの資源情報 (CPU 性能，HD 容量，IP アドレス) を分散的に管理する．サービスの負荷が高まった際には資源管理機構が利用可能な資源の中から要求を満たす資源を割当てる方式を用いている．本実行基盤により個人間で資源を融通し合うことが可能となり，計算機資源として共有される PC の管理や検索に対しスケーラビリティが得られるものと考える．

以下，2 節で提案実行基盤上で動作する PC グリッドの仕組みと実行基盤の概要について述べる．3 節で資源管理機構の詳細を説明し，4 節で提案実行基盤のプロトタイプの構成を示す．5 節では資源管理機構のシミュレーションと計算グリッドサービスの試作を行い，評価する．6 節は関連研究の紹介，7 節をまとめとする．

2. P2P グリッド実行基盤

2.1 P2P 環境での PC グリッド

本項では, 提案実行基盤上で動作する PC グリッドの仕組みと特徴について述べる $\big($図 $1(a)\big)$．本稿では提案実行基盤への参加者の PC が持つ計算機資源を参加者全員で共有し，計算機資源の提供と利用を相互に行う P2P 環境での PC グリッドを目指す．

リソースプール:　本実行基盤では参加者の CPU 性能と HD 容量を提供される計算機資源とする．CPU 性能は分散処理のための計算資源，HD 容量は分散ストレージのためのストレージ資源として利用される．地理的に分散した計算機資源を共有するため，提供された計算機資源を集積する仮想的な空間を用いる．本稿ではこれをリソースプールと呼ぶ．参加者から提供された計算機資源は一旦リソースプールに集められ，必要に応じて取り出し利用される．

サービス:　サービスとは本実行基盤で動作する計算グリッドや分散ストレージを指す．ネットワーク上に分散している計算機資源を利用するため，サービス利用者の PC とサービスに利用される計算機資源を持つ PC はオーバーレイネットワークを構成する．サービスはリソースプールから必要な計算機資源を取り出し利用する．

参加者が既存のサービスを利用する場合は自身の PC 上でもサービスを起動し，サービスのネットワークに参加する．もしサービスが存在しない場合，自身でサービスのネットワークを生成する．計算機資源をさらに必要とする場合，サービスはリソースプール上の計算機資源であるピアをサービスのネットワークに参加させる．サービスネットワークに参加しているピアが増えることでサービスが利用できる資源の総量も増加する．

本実行基盤では利用していないサービスのネットワークであっても資源として参加する場合がある．図 1(b) では，P_1，P_2，P_4 はサービス A のネットワークを構成している．一方，P_1，P_4 は P_3 とともに後述する資源管理のためのオーバーレイネットワークも構成している．多数の参加者が各自の目的に沿った形態で計算機資源を利用するために，本実行基盤上では複数のサービスを同時に動作させる．参加者は複数のサービスを同時に利用できる．

一般的に PC グリッドでは，構成 PC の頻繁な参加・離脱が想定される．本実行基盤ではサービスが計算機資源の利用主体であるため，サービスに参加している一部のピアが離脱してもサービス自体は他のピアにより存続する．ピアが離脱することによりサービスが利用可能な資源が減少しても，リソースプールから新たに資源を取得することでサービスを保つことができる．例えば図 1(b) においてサービス B は P_3，P_5，P_6 で分散的にデータを管理する分散ストレージサービスであるとする．仮に P_3 がサービス B から離脱しても，離脱の際にデータを P_5，P_6 に委譲を行えば，サービスで管理しているデータが失われることはない．

利用者の様々な要求に応じるため，本実行基盤は多種多様なサービスに対応可能であることが求められる．未知のサービスに対しても既存部分の変更を行わず利用できるようにするため，実装においての設計指針としてサービスを独立させる．サービスと実行基盤の核となるミドルウェア部分の依存関係がなるべく小さくするような設計とする．これにより利用者がそれぞれの目的にあったサービスを大きな制約を受けず作成できる．図 1(b) に示すようにサービスはそれぞれが独立したオーバーレイネットワークを構成するため，サービス毎に独自のネットワークトポロジやルーティング方式を設定できる．そのため，ネットワークの特性を生かしたサービスの開発が可能となる．

2.2 実行基盤の概要

前項で示した PC グリッドを実現するため，提案する実行基盤は計算機資源の管理やサービスへの計算機資源を割当てを行う．本稿では計算機資源の管理を行う仕組みを資源管理機構と呼ぶ．資源管理機構ではリソースプールに集積されている全参加ピアの資源情報を管理する必要がある．多数の参加者を想定している本実行基盤において資源管理機構には参加する PC 数の増加に対するスケーラビリティが必要である．そこで，資源管理機構を実行基盤における基本的なサービスと位置づけ，資源情報をデータとして P2P 方式で分散的に格納・抽出する分散ストレージとして構築する．本項ではまず資源管理機構と計算機資源である PC の関係を述べる．

2.2.1 P2P 方式の実行基盤

資源管理機構は参加ピアの中の任意のピアによる P2P 型分散ストレージとして構築される．分散ストレージを構成するピアを資源管理ピアと呼ぶ．全参加ピアは資源管理機構に自身の

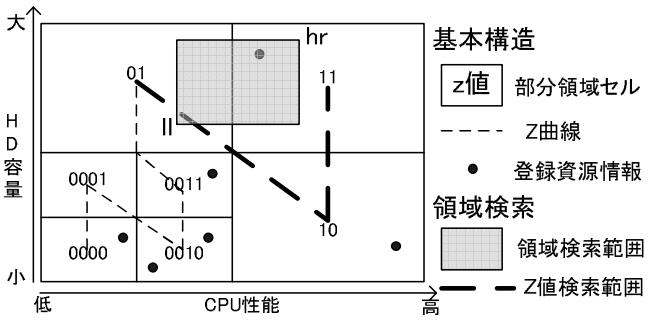

図 2　空間分割と索引付け，領域検索

Fig. 2　Space Division and Indexing, Range Search

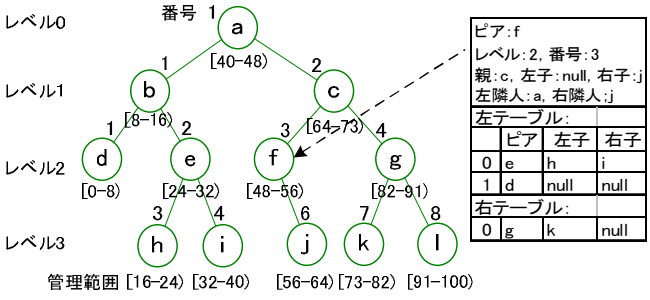

図 3　BATON 構造

Fig. 3　Structure of BATON

資源情報を登録する．そのため，各参加ピアには自身の情報を持つ資源管理ピアが必ず存在する．資源管理ピアは管理資源情報の登録ピアに，自身の IP アドレスを送信する．送信された IP アドレスにより，各参加ピアは自身の資源を登録した資源管理ピアとの間にリンクを張る．本稿ではピアが相互に IP アドレスを保持し，互いに通信可能である状態をリンクを張ることと定義する．図 1(b) に示すように資源管理ピアである P_1 が P_1,P_2 の資源情報を持つ場合，それらの間には資源登録によるリンクが張られる．資源管理機構は登録情報を P2P 方式で分散管理するため，資源管理ピアは自身の資源情報を持つとは限らない．資源管理ピア P_3 は自身の資源登録を P_4 に行っているため，資源を登録しているピアとしては P_4 にリンクを張る．

2.2.2　資源管理機構の利用

以下，本実行基盤において新規参加者による資源情報の登録と資源管理機構によるサービスへの資源割当について述べる．資源管理機構は資源情報の検索を行う機能を持つ．

資源登録:　本実行基盤への新規参加者はまず自身の PC 性能を資源情報として資源管理機構に登録する．新規参加者による資源登録の動作を以下に示す．

1. 新規参加者は既に参加しているピアに対しそのピアが資源登録によるリンクを張っている資源管理ピアを問合せる
2. 紹介された資源管理ピアを通じて資源管理機構へ新規参加者の PC 性能を資源情報として送信する
3. 資源管理機構によって決定した自身の資源情報を管理する資源管理ピアにリンクを張る

サービスが計算機資源を利用する場合，各ピアは資源登録によるリンクを用いて資源管理ピアに対し資源情報の更新を行う．

資源割当:　資源管理機構はサービスの稼働状況に応じて柔軟に資源割当を行う．サービスが必要としている計算機資源はサービスの種類や状況に依存するため，割当てる計算機資源はサービスからの要求に応じたものとする．サービスへの資源割当を以下に示す．

1. サービスは稼働状況に応じ資源管理機構に資源を要求
2. 資源管理機構は要求に応じた資源情報を検索
3. 検索された資源情報を資源要求を行ったサービスへ送信
4. サービスは送信された資源情報に基づきそのピアにサービスのオーバーレイネットワークへの参加を依頼
5. 参加依頼されたピアはサービスネットワークへ参加

3.　資源管理機構

3.1　機能と要件

資源管理機構は P2P グリッド実行基盤の基本サービスであり，資源情報を格納・抽出する機能を持つ．サービスが資源を要求する場合は，CPU 性能と HD 容量のペアをキーとして領域検索を行ない，要求を満足する参加ピアの情報を得る．

資源管理機構を P2P 方式で実現する際の要件を以下に示す．まず，サービスが要求する計算機資源の条件が単一とは限らないため，複数属性をキー値とした格納・検索の仕組みが必要である．また利用可能な資源と要求が完全一致しない場合に，要求に近い資源情報を検索するため範囲検索も必要となる．P2P 方式での実現を考慮した際には，検索クエリの到達性が保証されていることに加えクエリ転送負荷や応答時間を軽減するため，検索に要するホップ数がピア数 N に対して $O(\log N)$ 程度に抑えられることが望ましい．また，特定のピアに負荷が集中することを避け，ロードバランスを良好に保つ仕組みが必要である．

3.2　パラメータ空間の構成

管理機構は資源情報を配置するため，CPU 性能と HD 容量を座標系に用いた 2 次元パラメータ空間を生成し，P2P 方式による分散管理を行う．空間上に実行基盤参加ピアの CPU 性能,HD 容量を位置座標とした資源情報が写像される．CPU 性能を一元的に評価するにはベンチマークスコア等を用いる．

空間はピアの新規参加や負荷均衡処理の実行に合わせて，再帰的に 4 等分割される．分割の形状は空間全体を根とした 4 分木状になる．分割された部分領域には代表的な空間充填曲線である Z 曲線 [7] に従って ID(z 値) を与える．付与した z 値は空間索引として用いる．空間充填曲線による検索法は，木構造型索引の持つ根付近に検索処理が集中する問題が起こらない．空間分割の深さが一定ではない場合でも，Z 曲線ならば容易に ID 付けが可能である．空間を P2P 方式で分散管理する際は部分領域のセルが管理単位となる．

図 2 に 6 ピアの資源情報を配置した空間分割と索引付けの例を示す．全体領域が分割され更に左下の領域が再分割されている．各セルには分割の深さに応じた z 値を付与している．

3.3　ネットワーク構造

ピア間のオーバーレイネットワーク構造には BATON を用い，資源管理ピアがパラメータ空間を分散管理する．

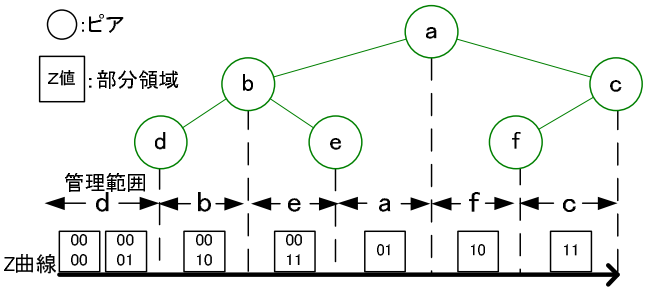

図 4　BATON 上での部分領域管理
Fig. 4　Partial region management on BATON

[BATON]

BATON は平衡 2 分木を基本構造としており木の節がピア，枝がリンクに該当する．代表的な P2P 構造の Chord [8] と比べ，BATON の長所としては各ピアの自律的な処理による負荷均衡機能と効率的な範囲検索が実現されている点が挙げられる．

各ピアは木の深さに対応したレベル，各レベル毎に左から順に付けられる番号を識別子として与えられる．親子間のリンクに加えて各ピアの左右に位置するピア間には隣人リンクを張る．更に各ピアは左右ルーティングテーブルを持ち，同レベルのピア間に Chord 状の遠隔リンクを張る．レベル L のピアはテーブルに計 L 個のリンクを保持する．番号 N のノードの右 (左) ルーティングテーブルの i 番目のリンク先は番号 $N+2^{i-1}(N-2^{i-1})$ のピアとなる．該当するピアが存在しない場合は空き状態とする．子を持つピアの左右ルーティングテーブルに空きが無ければ，平衡木であることが示されている．

図 3 にピア数 12 の BATON 構成例を示す．レベル 2, 番号 3 のピア f では左ルーティングテーブルに同レベルの番号が $3-2^0=2$, $3-2^1=1$ のピア e,d へのリンクを，右テーブルには番号 $3+2^0=4$ のピア g へのリンクを保持している．

3 種類のリンクを検索に使用することにより任意の地点へ $O(\log N)$ の検索ホップ数で到達可能であり，かつ木構造型ネットワークの欠点であるルート付近へのクエリの集中を防いでいる．新規ピアが参加する際は木のバランスが崩れない位置を検索し，その位置に加入する．また，ピアの離脱時にも木のバランスが崩れる可能性がある．その場合は葉ノードの中から離脱してもバランスが崩れないピアを検索し，そのピアと位置を交換した後，離脱する．各ピアは木全体の構造情報を持たないが，左右ルーティングテーブルの状態から処理の判断が可能である．

各ピアには隣人リンクに沿って連続的数値の管理範囲が割当てられ，管理範囲内に配置されたデータが格納される．データの配置にハッシュ関数等は用いず，キー値をそのまま格納位置とする．そのため，データの順序や位置関係が保存され範囲検索が容易に行える．負荷分散はハッシュに代わり，ピア間で自律的に負荷を調整する機能により実現される．

[部分領域セルの配置]

BATON ではピアに 1 次元の数値範囲を与えているが，提案機構ではセルが管理単位であるため z 値の範囲 [zMin,zMax] を管理範囲として与える．ここで zMin は各ピアが管理するセルを z 曲線に沿って昇順に並べた際に左端に位置する領域の z 値であり zMax は右端の z 値である．

Algorithm 1 A. 領域検索 (ll,hr,QR)

```
1: //ll,hr:初期値は検索領域の左下, 右上座標, QR:クエリ検索領域
2: zL ← getZID(ll)
3: zH ← getZID(hr)
4: if 自管理範囲 [zMin,zMax] と [zL,zH] に重なりがある then
5:     if 自管理領域が QR と重なっている then
6:         クエリ発信者に検索結果を返す
7:     end if
8:     if zMax∈[zL,zH] then
9:         zL ← QR と重なる最小の z 値 > zMax
10:        B ← A.FindCloser(zL)
11:        ll ← zL 対応領域の左下座標
12:        B. 領域検索 (ll,hr,QR)
13:    end if
14:    if zMin∈[zL,zH] then
15:        zH ← QR と重なる最大の z 値 < zMin
16:        B ← A.FindCloser(zH)
17:        ll ← zH 対応領域の右上座標
18:        B. 領域検索 (ll,hr,QR)
19:    end if
20: else
21:    B ← A.FindCloser(zL 又は zH)
22:    B. 領域検索 (ll,hr,QR)
23: end if
```

図 4 に図 2 のセルを BATON に配置した例を示す．zID:0000 から 11 までのセルが z 曲線に沿ってネットワーク上の左のピアから順に割当てられている．ピア d の場合，zMin=0000,zMax=0001 となりこの zID の範囲を管理する．

基本的なピアの参加・離脱・負荷均衡アルゴリズムは BATON に従い，管理範囲の受け渡し部分を提案機構に合わせて変更する．参加時に新規ピアの親となるピアは自身が管理するセルの半数をデータと共に新規ピアに委譲する．左の子の場合は左半分，右の子は右半分を委譲し，zMin 又は zMax を修正する．管理するセルが 1 つの際は，その領域セルを分割する．離脱時は全管理セルを親ピアに委譲する．

3.4　領域検索

資源管理機構を構成するピアはサービスの要求に従い，論理空間上に写像された資源情報から適合する情報を発見するため領域検索を行う．点検索や線分状の検索は領域検索の特殊な場合として処理可能である．

空間索引構造に従い，検索時は 2 次元の検索領域を 1 次元の z 値範囲 [zL,zH] に変換する．z 曲線による索引付けでは検索領域の左下座標の z 値が最小値 (zL)，右上が最大値 (zH) となる．z 値範囲に対応する領域は検索領域を包含する．範囲内には検索領域との重なりが存在しないセルも含まれるため，単一計算機での処理の場合は初めに重なりが存在するセルを計算し，クエリを複数の z 値範囲に分割する．しかし，提案機構では領域分割の深さが一定では無く，各ピアは領域全体の分割の形状を知らないため発信者が予めクエリを分割することができない．

上記の制約に対応するため, [9] で提案されている領域検索ア

Algorithm 2 A. 負荷均衡処理 ()

```
1: リンク先ピア情報から平均負荷を推定
2: if 自負荷 > 平均負荷 then
3:     ピア B ← 負荷の低い隣人ピア
4:     if B の負荷 < 平均負荷 then
5:         B と負荷を均衡する
6:     else if A が葉ノードである then
7:         C ← 左右ルーティングテーブル内で最も負荷の軽いピア
8:         C に再参加を依頼する
9:     end if
10: end if
```

ルゴリズムを利用する．アルゴリズムを Algorithm1 に示す．A が処理実行ピア，B が次に転送するピアである．クエリ発信者は検索領域の左下，右上座標を管理セルの分割レベルを基に z 値 (zL,zH) に変換し，[zL,zH] をクエリの z 値範囲として用いる．クエリを受け取ったピアは管理範囲がクエリ z 値範囲と重なっているならば，検索領域との重なりをチェックする．その後，自身の持つ情報から次に検索領域と重なる zL(zH) を計算し，クエリ z 値範囲を修正する．ピアの管理範囲が [zL,zH] に包含されている場合，クエリを 2 つに分割し z 値の増加・減少方向に転送する．getZID は任意の座標を自身の知る分割レベルで z 値に変換する処理，FindCloser はリンク先の中から検索値を超えない範囲で最も近いピアを選択する処理である．

図 2 の例では灰色の矩形が検索領域であり，z 値 01 から 11 が z 値検索範囲になる．z 値：01,10,11 に対応する領域が検索領域を包含していることが分かる．10 の領域は検索領域との重なりがないため，01 の管理ピアが始めにクエリを受け取った場合は次に 11 の管理ピアにクエリを転送する．11 の場合も同様に 01 に転送する．10 の場合はクエリを分割し，01 と 11 に転送する．各ピアは自身の持つ情報のみで転送先を判断するため，必要なピアを飛ばすことが起こりうるが，その場合でも転送先ピアがクエリを 2 方向に分割することで最終的には必要なピアにクエリが届く．

3.5 負荷均衡処理

ピアの負荷はデータ保持量やクエリ処理量等で測定され，各ピアはリンク先ピアの負荷情報を保持している．これらの負荷が特定のピアに集中することを防ぐため，負荷均衡処理を行う．具体的な負荷均衡は管理セルの受け渡しにより行う．

負荷均衡処理のアルゴリズムを Algorithm 2 に示す．まず，負荷均衡処理を行うピアはリンク先ピアの負荷情報から全体の負荷を推定する．自負荷が平均を超えているならば隣人ピアのうち，負荷の軽い側を均衡相手とする．その後，相手と負荷が均等になるまで管理セルを委譲する．管理セルが 1 つならば領域分割を行う．また，セルが委譲単位であるため 1 つのセルに大量のデータが格納されていると均衡量以上に相手に負荷を委譲してしまう可能性がある．よって予めセルに分割閾値を設定しておき，委譲セルに閾値を超えるデータが格納されている場合，分割して委譲単位を小さくする．

基本的な隣人間での均衡処理に加え，委譲相手も平均負荷を超えており，かつ自身が葉ノードである場合，左右ルーティングテーブルから最も負荷の軽いピアを選択し，自身の子として再参加させる処理を行う．全ピアが定期的に隣人間の均衡処理を実行することで負荷均衡は可能であるが，再参加処理により，負荷が均衡するまでの処理回数が抑えられる．また，再参加処理により木のバランスが崩れた場合，通常の平衡 2 分木と同様にローテーションを行いバランスを保つ．この際，ピアの木構造上の位置とリンクのみが変更され管理セルは変更されない．

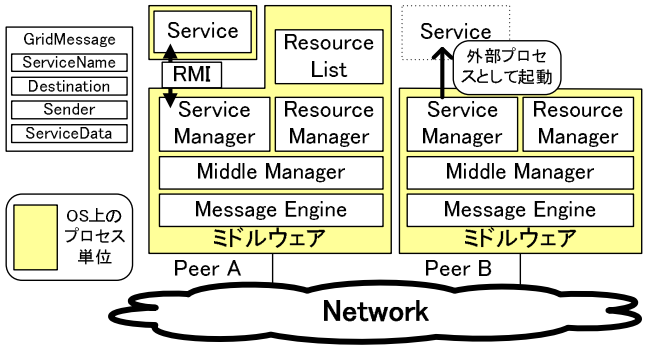

図 5　提案システムのコンポーネント構成
Fig. 5　Component structure of this system

4. 実　　装

本節では資源管理機構を備えた P2P グリッド実行基盤のためのミドルウェアの実装と計算サービスの試作について述べる．本実行基盤は Java 言語 (J2SE5.0) を用いて実装する．

4.1 設計指針

本稿では 2.1 項で述べたように実行基盤とサービスを独立させる．そのため実行基盤のためのミドルウェアとサービスを OS 上の別プロセスで動作するように実装する (図 5)．サブプロセス生成には Java 言語の ProcessBuilder クラスを利用する．

本実行基盤においてピアは計算機資源として未知のサービスネットワークへ参加を行なう場合がある．このときまずは既にサービスを起動しているピアからミドルウェアはサービスの実行に必要なバイトコードを受け取る．Java 言語を用いる現実装では異なる OS であっても同じバイトコードを用いてサービスを実行することが可能である．図 5 の Peer B のように ServiceManager よりプロセスとして起動する．

ミドルウェアは通信や資源割当の機能をサービスに提供する．サービスとミドルウェアは異なるプロセスで動作するため，サービスがミドルウェアの機能を利用するためには互いのプロセスで通信を行う必要がある．現実装ではミドルウェアとサービスの通信に Java 言語の RMI を用いる．RMI は異なる JavaVM 上のオブジェクトのメソッドを呼び出すことができる．このためサービスはミドルウェアと別プロセスで動作していてもミドルウェアが提供する機能を利用できる．

4.2 サービスの独立と実行基盤との連携

4.2.1 通信部分の共通化

現実装はサービスの開発を容易に行えるようにするため，通信用 API を用意しサービスに対し通信機能を提供する．本稿で

は異なるピア間の同じサービス同士でやり取りするメッセージをサービスメッセージと呼ぶ．サービスメッセージのピア間通信には図 5 の GridMessage クラスを用いる．GridMessage にはサービスの識別子である ServiceName，送信元，宛先のピア情報，サービスメッセージのデータを記述する．サービスメッセージのデータは Java 言語の直列化機能により GridMessage に格納する．これによりミドルウェアではサービスメッセージの形式を意識することなく通信が行うことができる．

ピア間通信は MessageEngine が担当する．MessageEngine は RMI を用いることで，全てのサービスが送信用 API を利用することを可能にする．また各サービスも受信用のリモートメソッドを実装し他のピアから MessageEngine が受信したメッセージを受け取れるようにしておく．

サービスメッセージのピア間通信における処理について述べる．まず送信側のピアのサービスでは GridMessage に ServiceName，宛先ピアの情報を記述する．次にサービスはサービスメッセージを直列化機能により GridMessage へ格納し，MessageEngine の送信用メソッドを呼び出す．MessageEngine では GridMessage に送信元として自身のピア情報を記述し，宛先ピアへ GridMessage を送信する．

受信側のピアでは，まず MessageEngine が GridMessage を受信する．その後 GridMessage は MiddleManager を経由し，ServiceManager へ送られる．ServiceManager では ServiceName によりどのサービス宛であるかが判定され，サービスの受信用リモートメソッドを利用して ServiceName が一致するサービスへ GridMessage が送信される．GridMessage を受け取ったサービスでは，復元化機能を用いてサービスメッセージのデータを抽出し通信を完了する．

4.2.2 資源管理機構の実装

現実装では 3 節で提案している BATON を用いた資源情報の分散管理手法は実装に至っていない．提案手法を用いた資源管理機構の実装は今後の課題である．

現実装でミドルウェアの ResourceManager はサービスからの資源要求に対して要求に応じた資源情報を返す機能を持つ．資源管理ピアは図 5 の Peer A のように ResourceManager と参加ピアの資源情報を格納した ResourceList を持ち，それ以外の参加ピアは Peer B のように ResourceManager のみを持つ．

サービスから資源要求が行われた場合，その要求は ServiceManager から MiddleManager を経由し，ResourceManager に伝えられる．資源管理ピアでない参加ピアは，資源登録によるリンクを用い資源管理ピアの ResourceManager へ要求を送信する．ResourceManager は資源要求に応じたピアの検索を行う．資源管理ピアの ResourceManager は送信された要求に応じたピア情報を ResourceList 内で探す．もし自身の ResourceList 内に要求に合うピア情報がない場合，他の資源管理ピアの ResourceManager に資源要求を送信する．

要求に応じた資源情報のピアを発見した場合，資源要求を発信したピアの ResourceManager へ資源情報が送られる．ResourceManager へ送られた資源情報は MiddleManager，ServiceManager を経由してサービスへ資源情報が伝えられる．

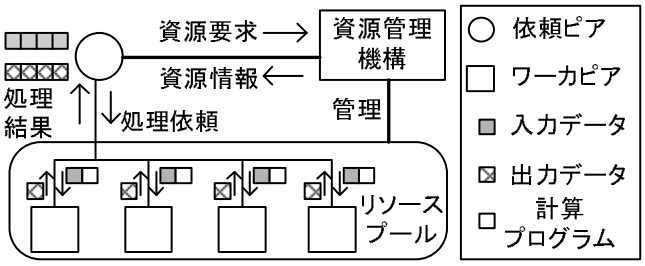

図 6　試作計算サービス
Fig. 6　Prototype of Processing Grid service

4.3 計算サービスの試作

本稿では実行基盤上で動作する計算グリッドの計算サービスの試作を行う．試作する計算サービスは計算タスクをマスタワーカ方式で並列処理する．計算処理を行いたいピアは試作サービスを起動してタスクの依頼ピアとなり，リソースプールにある計算資源をワーカとして利用する．計算サービスは以下の流れで処理を行う (図 6)．

1. 依頼ピアは資源管理機構へ PC 台数による資源要求送信
2. 資源管理機構は要求の数の資源情報を依頼ピアへ送信
3. 依頼ピアは送信された資源情報の数に入力データを等分割
4. 依頼ピアはワーカピアへ分割された入力データと計算プログラムを計算依頼として送信
5. ワーカピアは受信した入力データを計算プログラムを用いて処理を行い，終了すると依頼ピアへ出力データを返信
6. 依頼ピアは全ワーカピアから計算結果が返信されるまで待ち，出力データが全て揃うとサービスを終了

試作する計算サービスは，パターンマッチングやモンテカルロ法のように入力データのみで計算処理ができ，処理や出力データに依存関係がないものを想定している．

5. 評　　価

5.1 資源管理機構シミュレーション

シミュレータにより資源管理機構の検索処理と負荷均衡処理を評価する．データを配置する論理空間は倍精度の $[0, 500] \times [0, 500]$ の正方領域に設定した．

検索処理：検索処理に関するスケーラビリティを評価するため，点検索と領域検索に要するホップ数を計測した．点検索はデータ数を 100000 で固定し，全ピアが 10 回ずつランダムな位置を検索する設定でピア数を 1000-10000 まで変化させピア毎の平均ホップ数を計測した．領域検索は点検索と同様の設定で 50×50 のサイズのランダムな位置に対するクエリを生成し，クエリが分割された場合は最大値をその検索のホップ数とした．

計測結果の平均値と最大値を図 7 に示す．図より点検索ではピア数の増加に対する平均ホップ数の増加が抑えられていることが分かる．ピア数 1000 の場合と 10000 の場合の差は約 2.3 である．また，最大ホップ数にも大きな増加は見られない．

領域検索に関しては，平均ホップ数が約 12 から約 26 に，最大ホップ数は 18 から 37 に増加している．点検索に比べて増加の割合が上昇している理由として検索領域の大きさを固定して

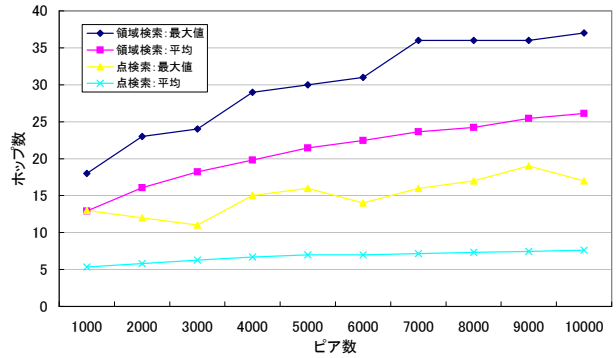

図 7　ピア数に対する検索ホップ数平均と最大値

Fig. 7　Average and Maximum of hop counts for query to the number of peers

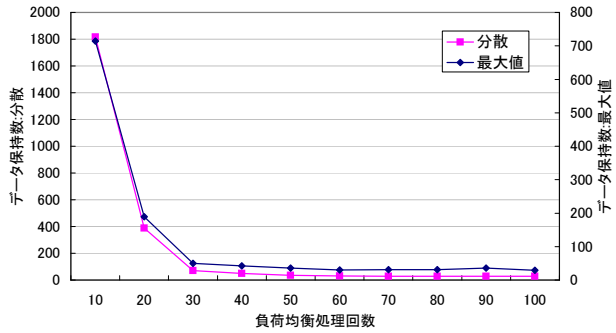

図 8　負荷均衡処理時のデータ保持数分散と最大値：一様分布

Fig. 8　Variance and Maximum of data size by loadbalancing:Uniform data distribution

いるため，ピア数の増加に伴い検索領域内のピア数が増加したことも影響していると考えられる．しかしながら領域検索の場合もピア数の増加に対してホップ数の増加は抑えられている．

負荷均衡処理：負荷均衡処理に伴うロードバランスの向上を評価するため，データ配置が一様な場合と偏りがある場合を想定し，シミュレーションを行った．ここでピアの負荷はデータ保持数とする．前者は乱数により領域全体にランダムにデータを配置し，後者は [0,10]×[0,10] の範囲に集中させた．双方ともピア数 10000, データ数 100000，分割閾値 5 で実行し，データ保持数を計測した．シミュレーション結果を図 8,9 に示す．ここで均衡処理回数は各ピアがアルゴリズムを実行した回数であり，再参加が発生した場合はそれが終了するまでを 1 回としている．

データ配置が一様な場合，60 回程度実行することで最大値が約 700 から 30 に，分散が約 1800 から 30 まで下がった．その後は最大値には大きな変化が見られず，分散は微減を続けている．実行結果より，ピア数に対して少ない回数の処理で負荷が均衡されることが分かる．

データ配置を偏向させた場合は，負荷均衡処理前の時点で BATON 上の左端のピアがほぼ全てのデータを保持しており，負荷均衡に必要な処理回数が最も多くなると考えられる．図 9 に示す結果では，負荷均衡処理を 5000 回実行することで最大値が 32，分散が 28 まで減少した．一様な配置に比べ処理回数は増加するが，処理に伴って確実にロードバランスが向上しており，1 万程度のピア数ならば最悪の場合でもピア数の半分程度の処理回数で良好な状態になることが分かる．

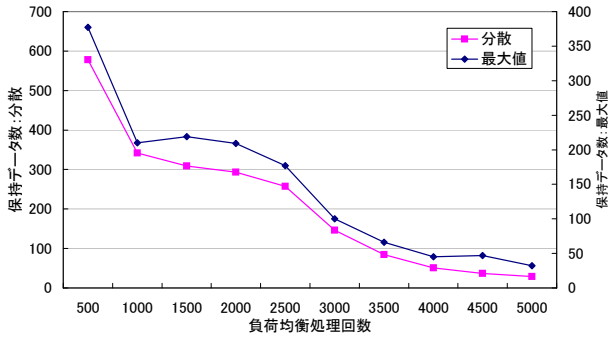

図 9　負荷均衡処理時のデータ保持数分散と最大値：偏向分布

Fig. 9　Variance and Maximum of data size by loadbalancing:Skewed data distribution

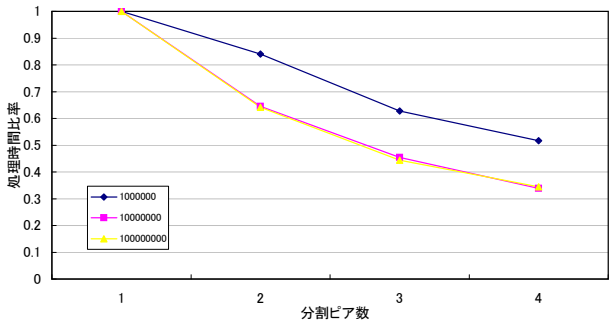

図 10　分割ピア数による処理時間比率の変化

Fig. 10　Ratio of processing time with respect to the number of peers

表 1　分割ピア数に対する処理時間

Table 1　Processing time with respect to the number of peers

	分割ピア数			
	1	2	3	4
2—10^6	2.0 秒	1.7 秒	1.2 秒	1.0 秒
2—10^7	40.8 秒	26.3 秒	18.6 秒	13.8 秒
2—10^8	1137.0 秒	729.2 秒	504.2 秒	392.2 秒

5.2　計算グリッドサービス

本実行基盤の資源割当を用いて試作した計算グリッドの動作検証を行う．検証では試作した計算サービスのタスクとして素数判定を行う．素数は暗号化に利用され，素数発見のための分散コンピューティングプロジェクトが存在する [10].

今回は判定対象である整数 n に対し，$m \in M = \{2\} \cup \{2k+1 \mid k \in \mathbf{N}, k \geqq 1, 2k+1 \leqq \sqrt{n}\}$ である全ての m で $n \equiv 0 \pmod{m}$ となるかの判定を行う．計算サービスでは n を 2 から 10^6，10^7，10^8 の間を判定範囲として変化させ，繰り返し素数判定を行い素数である整数を列挙する．この判定方式は入力データである n のみで素数判定処理が行えるため，並列化が可能で試作した計算サービスで処理するのに適している．

実験は WindowsXP，CeleronM 1.73GHz，HD 容量 40GB の PC を 4 台 LAN で接続し行う．資源管理機構へ要求する PC 台数を変化させ，資源要求から素数判定の終了までに要した処理時間を計測する (表 1).

1 台の場合の処理時間を 100 %とした時の処理時間の比率を図 10 に示す．全ての場合において，分割ピア数を増やしたことで処理時間の短縮が見られる．これは判定範囲をピアの数で

分割するため分割数が増えるほど 1 ピアが処理するタスクが小さくなったと考えられる．

判定範囲が 2 から 10^6 は他の判定範囲に比べて，分割ピア数の増加による処理時間の短縮が大きく見られない．これは表 1 に示すように 2 から 10^6 の場合ごく短時間に処理が完了しており，素数判定に要した時間に対して資源要求やピア間での通信に掛かる時間が大きく影響していると考えられる．

タスクの大きさにより処理時間の短縮に差はあったが，全ての場合において処理時間が短縮しており，本実行基盤が分散処理を行う計算グリッドのために機能していることを示す．

6. 関 連 研 究

P2P による代表的なデータの分散管理手法には DHT(Distributed Hash Table) として知られる Chord,Pastry [11] 等がある．DHT はハッシュ関数の乱数性に基づく負荷均衡と $O(\log N)$ ホップでの完全一致検索を実現している．しかしこれらの手法はハッシュ関数によりデータ配置を決定するため，データの順序や類似性が格納位置に反映されない．そのため範囲検索が困難である．また，スキップリストの構造を P2P に応用した手法として SkipNet [12] が提案されている．ピア間のリンク構造は確率的に決定され，$O(\log N)$ での完全一致検索に加え範囲検索を実現している．しかし，保持データに関するシステム全体の負荷均衡は保障されていない．

取り扱うデータの属性を多次元に拡張した手法では [13] [14] [15] が提案されている. [13] [14] は地図のような空間的データを扱うことを目的とし，木構造状に分割された 2 次元論理空間を DHT 上に配置している．木構造に沿ったルーティングを行うことで DHT の機能に加えて領域検索が実現されている．根付近に負荷が集中する問題にも対策が取られているが，完全には解消されていない. [15] は空間を R 木状に分散管理する手法である．各ピアは管理領域を基に R 木状に階層的グループを形成し，空間上の任意の範囲を検索可能である．１つのピアが複数の管理領域を持つ場合や複数のグループと交差するクエリについては対応していない．

PC グリッドに関しては [3] や [10] 等，余剰リソースを集めて使用する様々な研究が行われてきた．しかしこれらの研究はセンタサーバが参加 PC を管理する方式であるためスケーラビリティに問題があり，PC のみでシステムを構築することは難しい．そこで，グリッド資源情報の分散的な共有方式が提案されている [16] [5]. [16] は Chord を用いてグリッド資源情報を共有する手法で，複数の資源情報データをハッシュ関数により同一のネットワーク上に配置する．局所性保存ハッシュ関数により，データの数値属性順序を保存し，１次元範囲検索を可能にしている．しかし，元々のデータの偏りも保存されるためデータの分布によっては特定のピアに負荷が集中する可能性がある. [5] は市場原理に基づいたグリッド資源共有方式である．リソース情報の登録と検索はマーケットブローカと呼ばれるサーバ群で分散的に実行される．検索は複製を配置することで実現される仕組みであり，ストレージ資源の総消費が大きい．また，登録データ分布の偏りに対しての負荷分散も課題とされている．

7. おわりに

P2P 環境における PC グリッド実行基盤を提案し試作による評価を行った．シミュレーションにより資源管理機構の検索と負荷均衡処理に関するスケーラビリティ，実機環境での実験で提案実行基盤及び試作した計算サービスの有効性を確認した．

本稿において試作した計算サービスでは判定範囲を等分割し，各ワーカピアへ処理を依頼する方式を用いた．しかし今回用いた素数判定の場合，判定対象の整数が大きくなるほど判定に要する処理は大きくなることが考えられる．このような場合に対して動的にタスクを分割し配分する手法が必要となる．また，処理時間に与える通信遅延の影響を考慮する必要がある．多数の PC へ資源管理機構を実装した場合の検索や負荷均衡処理の評価や計算サービスの評価も今後の課題である．

文　　献

[1] “グリッドコンピューティング特集”, 情報処理学会誌, vol.44, no.6, pp.574–614, 2003.

[2] “ビジネスグリッドコンピューティング特集”, 情報処理学会誌, vol.47, no.9, pp.946–985, 2006.

[3] SETI@home. http://setiathome.ssl.berkeley.edu/.

[4] cell computing. http://www.cellcomputing.jp/.

[5] 玉井森彦, 柴田直樹, 安本慶一, 伊藤実, “PC グリッド環境での市場原理に基づいた資源共有方式”, 情報処理学会論文誌, vol.47, no.2, pp.455–464, February 2006.

[6] H.V. Jagadish, B.C. Ooi, and Q.H. Vu, “BATON: A alanced tree structure for peer-to-peer networks”, In Proceedings of the 31st VLDB Conference, 2005.

[7] J.A. Orenstein and T.H. Merrett, “A class of data structures for associative searching”, In Proceedings of the 3rd ACM SIGMOD PODS, pp.181–190, 1984.

[8] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, “Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications”, In Proceedings of the ACM SIGCOMM’01, pp.149–160, 2001.

[9] Y. Shu, B.C. Ooi, K.L. Tan, and A. Zhou, “Supporting Multi-dimensional Range Queries in Peer-to-Peer Systems”, In Proceedings of the Fifth IEEE International Conference on Peer-to-Peer Computing (P2P’05), pp.173 – 180, 2005.

[10] G.I.M.P. Search. http://www.mersenne.org/prime.htm.

[11] A. Rowstrom and P. Druschel, “Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems”, In Proceedings of Middleware’01, November 2001.

[12] N.J.A. Harvey, M.B. Jones, S. Saroiu, M. Theimer, and A. Wolman, “SkipNet: A Scalable Overlay Network with Practical Locality Properties”, In Proceedings of 4th USITS, March 2003.

[13] E. Tanin, A. Harwood, and H. Samet, “A Distributed Quadtree Index for Peer-to-Peer Settings”, In Proceedings of the IEEE ICDE’05, pp.254–255, 2005.

[14] Y. Chawathe, S. Ramabhadran, S. Ratnasamy, A. LaMarca, S. Shenker, and J. Hellerstein, “A case study in building layered DHT applications”, In Proceedings of the ACM SIGCOMM’05, pp.97 – 108, August 2005.

[15] A. Mondal, Yilifu, and M. Kitsuregawa, “P2PR-tree: An R-tree-based Spatial Index for Peer-to-Peer Environments”, In Proceedings of the EDBT’04, pp.516–525, 2004.

[16] M. Cai, M. Frank, J. Chen, and P. Szekely, “MAAN: A Multi-Attribute Addressable Network for Grid Information Services”, Journal of Grid Computing, vol.2, pp.3–14, 2004.