資源管理機構を持つ P2P グリッド実行基盤の検討と試作

Prototyping and Evaluation of P2P Grid Platform with Resource Management Mechanism

加藤 宏章 ♥ 水江 真登 ♦
廣川 雅基 ♠ 小林 孝史 ♣
上島 紳一 †

Hiroaki KATO Masato MIZUE
Masaki HIROKAWA Takashi KOBAYASHI
Shinichi UESHIMA

近年, グリッドコンピューティング技術や 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．提案実行基盤では分散的に計算機資源を管理するため，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．ここでは，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び，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と評価を行う．

Researches on Grid computing and P2P systems are gaining focuses recently. This paper proposes a PC grid platform in P2P settings. In this platform, PC's are managed distributively in a P2P manner by the resource management mechanism proposed in this paper. Peers on the 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 observe the effects on the resource management mechanism.

♥ 正会員 関西大学総合情報学部総合情報学科 hir.kato@gmail.com
♦ 学生会員 関西大学大学院総合情報学研究科 fb5m130@edu.kansai-u.ac.jp
♠ 学生会員 関西大学大学院総合情報学研究科 menhaoran@gmail.com
♣ 非会員 関西大学大学院総合情報学研究科 kobayasi@res.kutc.kansai-u.ac.jp
† 正会員 関西大学大学院総合情報学研究科 ueshima@res.kutc.kansai-u.ac.jp

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]．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 [2]．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 [3].

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，参加ピアの資源情報を分散的に管理する．P2P 方式を用いた本実行基盤により個人間で資源を融通し合うことにより，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．

2. P2P グリッド実行基盤

2.1 P2P 方式を用いた実行基盤の構成

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 (図 1(*a*)).

リソースプール: 本実行基盤ではまず参加者の CPU 性能と HD 容量を提供される計算機資源として考える．ここでは提供された計算機資源を集積する論理空間を用いる．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．もし必要な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．

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．

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

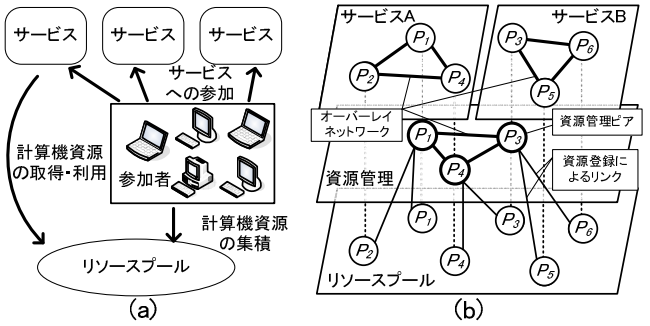

図 1 (a) 提案実行基盤上の PC グリッド，(b) ネットワークの多重構造

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

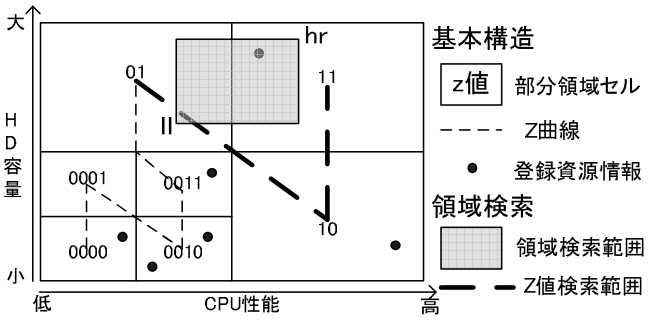

図 2 空間分割と索引付け，領域検索

Fig. 2 Space Division and Indexing, Range Search

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

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び，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．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する．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資源情報を登録する．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．

3. 資源管理機構

3.1 機能と要件

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，負荷均衡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，参加する PC の性能値を座標系に用いた n 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．本稿では簡単のため CPU 性能と HD 容量のみを座標系に用いた 2 次元パラメータ空間について議論するが，一般の n 次元についても以下に述べる Z 曲線の性質により同様に扱うことができる．実際には PC の性能としては，主記憶容量，通信速度，ディスク速度等がある．

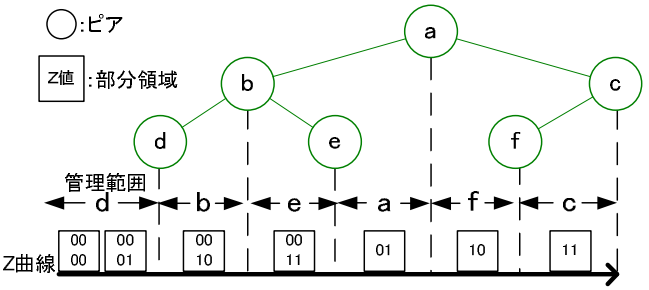

図 3 BATON 上での部分領域管理

Fig. 3 Partial region management with BATON

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 [4] 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．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．

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[5] を用い，資源管理ピアがパラメータ空間を分散管理する．

[部分領域セルの配置]

図 3 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，0000 と 0001 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．

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．

空間索引構造に従い，検索時は n 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，[6] で提案されている領域検索アルゴリズムを利用する．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．

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，管理セルの受け渡しにより負荷均衡処理を行う．

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．

4. 評価

4.1 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．

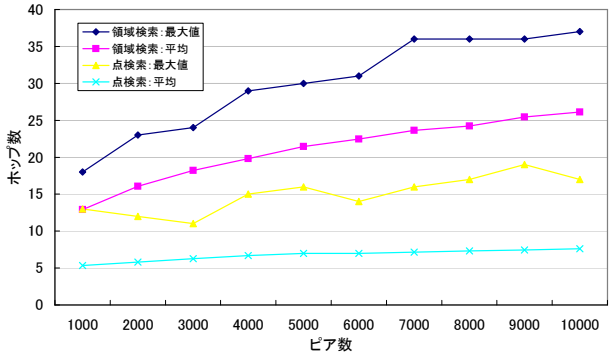

図 4 ピア数に対する検索ホップ数平均と最大値

Fig. 4 Average and Maximum of hop counts for query to the number of peers

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 4 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．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，点検索に比べて増加の割合が上昇している．理由としては検索領域の大きさを固定しているため，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負荷均衡の向上を評価するため，データ配置が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．データは $[0,10] \times [0,10]$ の範囲に集中させた．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 5 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．

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 5 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．処理に伴って確実に負荷均衡が向上しており，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．

4.2 パラメータ数の増加による影響

このシミュレーションの評価は CPU 性能，HD 容量の 2 次元パラメータ空間の場合の結果である．パラメータの数（論理空間の次元数）が増加した場合にも同様に対応することができる．

パラメータ空間の次元数が n 次元になった場合も，同様に Z

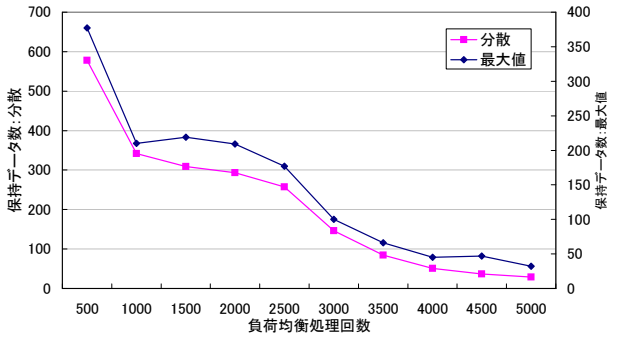

図 5　負荷均衡処理時のデータ保持数分散と最大値：偏向分布

Fig. 5　Variance and Maximum of data size by loadbalancing:Skewed data distribution

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

曲線を用いて空間を 1 次元的に順番付けるため [6]，ホップ数は次元数に独立である．領域検索については，次元数が増加するとホップ数も増加するが，本機構で利用する場合，PC の各性能をパラメータとして用いるので次元数が爆発的に増えるということは考えられない．よって範囲検索のホップ数も実用的な範囲に収まると考えられる．したがって，本機構は，次元数が増えた場合でも十分に実用に堪えうると考えられる．

4.3　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．ここ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．即ち，判定対象である整数 n に対し，$m \in M = \{2\} \cup \{2k+1 \mid k \in \mathbf{N}, k \geq 1, 2k+1 \leq \sqrt{n}\}$ である全ての m で $n \equiv 0 \pmod{m}$ 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．

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)．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．

5.　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．

今後の課題として，動的なタスクの分割・配分，処理時間に与える通信遅延の影響を考慮する等があげられる．また多数の 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] “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.

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

[3] 玉井森彦, 柴田直樹, 安本慶一, and 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.

[4] 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.

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

[6] 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.

加藤 宏章　Hiroaki KATO

2007 関西大学総合情報学部総合情報学科卒業．分散コンピューティングの研究・開発に従事．日本データベース学会正会員．

水江 真登　Masato MIZUE

2007 関西大学大学院総合情報学研究科博士課程前期課程修了．グリッドコンピューティングに関する研究・開発に従事．現在は日本電気株式会社勤務．日本データベース学会学生会員．

廣川 雅基　Masaki HIROKAWA

2007 関西大学大学院総合情報学研究科博士課程前期課程修了．P2P システムの研究・開発に従事．現在は日本電気株式会社に勤務．日本データベース学会学生会員

小林 孝史　Takashi KOBAYASHI

関西大学大学院総合情報学研究科准教授．1994 関西大学大学院工学研究科博士課程前期課程修了．マルチメディア情報システム，セキュリティシステムの研究・開発に従事．電子情報通信学会，人工知能学会などの会員．

上島 紳一　Shinichi UESHIMA

関西大学大学院総合情報学研究科教授．1983 京都大学大学院工学研究科博士課程単位取得退学，工学博士．マルチメディア情報システムの研究・開発に従事．情報処理学会，日本データベース学会などの会員．