情報処理学会論文誌 Vol. 52 No. 2 401–414 (Feb. 2011)

# 動的負荷分散機能を持つ高性能ボランティアコンピューティングの実現

村田善智[†1] 石杜佑記[†2]
滝沢寛之[†2] 小林広明[†1]

遊休計算資源を活用するボランティアコンピューティングは，単体では実現困難な巨大な計算能力を安価に実現可能なことから，次世代の高性能計算基盤として着目されている．しかし，遊休計算資源が提供する計算能力が動的に変動するため，ボランティアコンピューティングでは一定の計算能力を分散計算に提供し続けることを保証できない．本論文は，安定した計算能力を提供する高性能ボランティアコンピューティングを実現するために，P2P 型動的タスクスケジューリング機構を提案する．提案するタスクスケジューリング機構は，動的負荷分散とプロキシダウンロードの 2 つの機能を持つ．動的負荷分散は，遊休計算資源の予測不可能な負荷変動による計算時間の増加の問題を解決する．プロキシダウンロードは，遊休計算資源がサーバからのタスク取得に失敗したときの代替手段を提供し，サーバに生じるアクセス集中を回避する．提案するタスクスケジューリング機構を組み込んだボランティアコンピューティングを用いて実計算資源を集約し，提案手法の有効性を評価する．評価結果では，90%以上の実効効率を実現し，提案手法がボランティアコンピューティングの性能を改善し，高性能な大規模分散計算を実現できることが示された．

# A High-performance Volunteer Computing Environment with a Dynamic Load-balancing Mechanism

Yoshitomo Murata,[†1] Yuki Ishimori,[†2]
Hiroyuki Takizawa[†2] and Hiroaki Kobayashi[†1]

The volunteer computing is a new and promising style of large-scale distributed computing, which uses idle computing resources of individuals. However, the volunteer computing environment cannot guarantee to achieve a certain processing speed. This paper proposes a P2P-based dynamic task scheduling mechanism to increase the efficiency of volunteer computing. The proposed mechanism provides two features: decentralized load balancing and proxy download. The former feature reduces the variation of the execution times for individual tasks, which are usually aggravated by dynamic and unpredictable load changes on volunteer computing resources. The latter offers another way to assign tasks to idle computing resources when the server fails in the task assignment. By using two kind of distributed computing testbed, this paper examines the effect of the proposed mechanism. The experimental results show that the proposed mechanism achieves large scale distributed computing at an execution efficiency of more than 90%.

## 1. はじめに

民生用コンピュータの高性能化と低価格化により，高性能な個人用コンピュータやゲーム機器が一般家庭に広く普及している．しかし，それらの計算資源はつねに利用されているわけではなく，稼働時間の大半は遊休状態となっている．そこで，これらの遊休状態にある計算資源の処理能力を集約して大規模科学技術計算を行う，ボランティアコンピューティング[1)] が提案されている．実際，SETI@home[2)] や Folding@home[3)] などのボランティアコンピューティングプロジェクトでは，数 PFlop/s を超える処理性能を実現している．

このボランティアコンピューティングを計算基盤として活用するアプリケーション例として，大規模分散データマイニングがあげられる[4),5)]．情報社会の発展にともなう膨大なデータ生成により，データマイニング処理を実用的な時間内で完了可能な高い計算能力が求められている．ボランティアコンピューティングは大量の計算能力を安価に集約することが可能であり，情報爆発時代における大規模データマイニングの実現手段として非常に期待されている．

一方，ボランティアコンピューティング環境では，集約された計算資源の利用効率が低下しやすいという問題点がある．ボランティアコンピューティングでは，計算資源の所有者によって実行されるローカルのプロセスによって，利用可能な計算性能が動的に変化する．そのため，ボランティアコンピューティングでは，個々のタスクにつねに一定の計算時間が割り当てられることが保証されていない．Casanova らは文献 6)–8) において，計算資源の遊休・非遊休状態や分散計算に提供可能な処理能力などがつねに変動する環境におけるタスクスケジューリングについて述べている．その中で，各計算資源の状態を監視し，発生

†1 東北大学サイバーサイエンスセンター
Cyberscience Center, Tohoku University
†2 東北大学大学院情報科学研究科
Graduate School of Information Sciences, Tohoku University

した動的変動に応じてスケジューリングを再度行うことで負荷分散を行うことが有効であると述べた．一方，Casanova らの手法では，専用のスケジューリングサーバが計算資源の動的情報を一元的に取り扱う，マスター—ワーカ型の集中管理モデルを用いている．しかし，ボランティアコンピューティングでは広域ネットワーク上の多数の計算資源を対象としたスケジューリングを行わなければならず，スケジューリングに長い処理時間を必要とする．また，集中型の手法では負荷の集中や単一故障点の問題が生じてしまい，ボランティアコンピューティングにおいて適切なスケジューリングを行うことは困難である．また，多数のクライアントが参加する大規模なボランティアコンピューティング環境においては，複数のクライアントが 1 つのサーバに同時にアクセスする可能性が高まるため，クライアントからのアクセスに対するサーバの応答性能が低下することも問題となる．

計算資源が提供する処理能力が動的に変動する分散計算環境における動的負荷分散手法として，我々はこれまでに分散協調型スケジューリング機構を提案し，シミュレーションによる評価を行ってきた[9]．分散協調型スケジューリング機構では，単一のサーバが行っていた負荷分散処理を並列化し各計算資源で分散して処理することで，従来手法に比べ高いスケーラビリティを実現する．シミュレーションによる評価結果では，従来手法では計算資源の台数増加により動的負荷分散が有効に機能しなくなったのに対し，分散協調型スケジューリング機構を用いた手法では計算資源の台数に依存せずつねに動的負荷変動に対して有効な負荷分散を実現することが示されている．

一方，文献 9) では動的負荷分散機構のみが取り上げられており，タスクのクライアントへの分配機構など分散計算システムを実現するための具体的な仕組みについては述べられていない．本論文では，クライアントの稼働率が高く，高速処理が可能なボランティアコンピューティング環境の実現を目標とし，ボランティアコンピューティングの基盤ミドルウェアである *Berkeley Open Infrastructure for Network Computing*（以下 BOINC）を基盤とする分散計算システムのための，分散協調型スケジューリング機構に基づく動的負荷分散を提案する．さらに本論文では，この動的負荷分散機構に加えて，分散協調型スケジューリングを用いた新たなタスク配布手法の提案を行い，大規模環境におけるサーバへのアクセス集中による性能低下回避を実現する．分散協調型スケジューリング機構を BOINC と組み合わせることにより，BOINC が持つボランティアコンピューティング環境構築能力を維持したまま，より高性能なボランティアコンピューティング環境を容易に実現することができる．さらに，広域分散 PC クラスタや家庭用ゲーム機を用いた実証実験環境の構築を行い，実データを用いた大規模分散データマイニングを用いた評価実験を通じて，実環境での提案手法の有効性を明らかにする．

以下，2 章では，ボランティアコンピューティングを実現する基盤ミドルウェアである BOINC について説明し，計算資源上での動的負荷変動とサーバへのアクセス集中によってシステム全体の性能が著しく低下することを述べる．3 章では，ボランティアコンピューティングの性能を向上させるための仕組みとして，動的負荷分散機構と，プロキシダウンロード機構を提案する．4 章では，実計算資源を用いた分散データマイニング環境を構築し，実科学技術データの解析による提案手法の有効性の評価を行う．最後に，5 章で本論文のまとめを行う．

## 2. ボランティアコンピューティング

### 2.1 BOINC プラットフォーム

BOINC[10] は，現在最も一般的に利用されているボランティアコンピューティングミドルウェアである．現在，BOINC をプラットフォームとした数多くの分散計算プロジェクトが進行している[2),3),11)–18)]．

BOINC の概略を**図 1** に示す．BOINC のプラットフォームは，ボランティアコンピューティングプロジェクト全体の管理のために用いられる BOINC プロジェクトサーバと，実際に計算能力をプロジェクトに提供する複数の BOINC クライアント（以下，クライアン

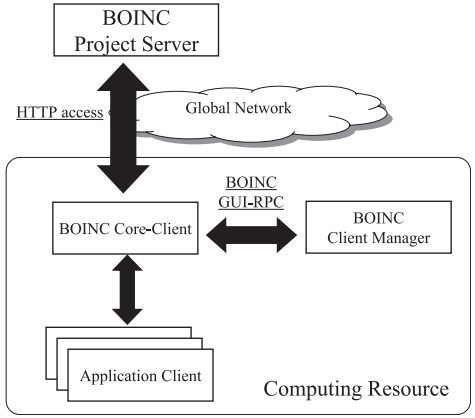


**図 1**　BOINC 概略
Fig. 1　Overview of BOINC.

ト）で構成されている．また，BOINC プロジェクトサーバとクライアント間の通信には，HTTP が用いられている．

BOINC プロジェクトサーバは，ウェブインタフェース，タスクサーバ，データサーバ，データベースサーバそしてバックエンドプログラムで構成される[1)]．ボランティアコンピューティング上で処理されるタスクの最小単位はワークユニット（以下，WU）と呼ばれ，プロジェクトの WU はプロジェクトサーバのデータベース内に格納される．BOINC プロジェクトサーバは，クライアントからの WU 要求を受けると，データベースから未処理の WU を取り出し，クライアントへ HTTP を用いて送信する．また，クライアントによる WU 処理が終了すると，クライアントは HTTP を用いて計算結果を BOINC プロジェクトサーバへアップロードし，最終的にデータベースへ計算結果が格納される．

各クライアントは，コアクライアント，アプリケーションクライアント，そしてクライアント管理プログラムで構成される．コアクライアントは BOINC クライアントの中核となるプログラムであり，BOINC プロジェクトサーバから WU のダウンロード，および計算結果のアップロードを行う．アプリケーションクライアントは実際に WU の計算を行うプログラムであり，コアクライアントによって実行される．クライアント管理プログラムは，計算資源の所有者によるクライアントの管理に用いられる．また，コアクライアントとクライアント管理プログラムは，BOINC GUI-RPC と名付けられたプロトコルに基づいて通信を行う．

BOINC を用いた分散計算の処理手順を以下に示す．プロジェクトの管理者は，ボランティアコンピューティングで計算したい WU をプロジェクトサーバのデータベースに登録する．一方計算資源の所有者は，自分の計算資源にクライアントをインストールし，計算資源を提供したいボランティアコンピューティングプロジェクトへ登録する．プロジェクトへの登録が完了すると，コアクライアントはプロジェクトサーバからアプリケーションプログラムと WU を自動的にダウンロードする．計算資源は，余剰計算能力を用いてタスクを処理し，処理結果をプロジェクトサーバにアップロードするとともに，必要に応じて新たな WU をダウンロードする．

### 2.2　性能低下の要因

BOINC を用いたボランティアコンピューティングにおいて，計算資源が提供する計算能力が動的に変動することによって生じる問題を以下に述べる．

#### 2.2.1　ターンアラウンドタイムの増加

プロジェクトサーバの負荷を軽減しボランティアコンピューティングの計算性能を高めるために，プロジェクトサーバはクライアントへ 2 つ以上の WU を 1 度に送信する．しかし実際の環境では，クライアントの負荷が動的に変動したり，シャットダウンによるボランティアコンピューティングプロジェクトからの離脱が生じる．そのため，あるクライアントにおける WU の計算時間を見積もることは非常に困難である．また，クライアントが WU を保持したままプロジェクトから離脱してしまい，計算結果が返却されないという問題がある．

計算結果が返却されなくなるという問題に対し，BOINC では WU に計算結果を返却するまでの期限を設ける．この期限を過ぎても計算結果が返却されない場合，プロジェクトサーバはその WU の処理が失敗したと見なし，WU を違うクライアントに再割当てする．しかし，再割当てが行われるとそれまでに WU に費やした処理能力が無駄になるうえに，再割当て前の分を考慮した WU の総ターンアラウンドタイムが増大し，ボランティアコンピューティングの計算性能が低下する．計算資源を効率的に利用するためには，WU の処理進捗を常時監視し，計算時間の増大に対し即座に WU の再割当てを行う必要がある．

#### 2.2.2　WU ダウンロードの失敗

BOINC プロジェクトサーバに対し複数のクライアントが同時にアクセスすることで，負荷の集中による WU ダウンロードの失敗が起こる．クライアントが WU のダウンロードに失敗した場合，そのクライアントでは計算が行われず，計算に利用できたはずの処理サイクルが無駄になる．その結果，ボランティアコンピューティングプロジェクトのクライアントの利用効率が低下し，本来の計算性能を発揮できない．

また，ボランティアコンピューティング環境を用いてデータマイニングを行う場合，WU には大きなデータファイルが添付される．そのため，より多くクライアントを用いて高速な計算を行おうとすると，サーバからクライアントへの WU 転送能力がボトルネックとなる．WU 転送数がサーバが処理可能な限界を超えた場合，クライアントを増やしても WU のダウンロードが行われず，システム全体の性能は向上しない．

ボランティアコンピューティングにおいて，計算資源の計算能力を最大限引き出すためには，WU ダウンロードの失敗を防ぎ，また失敗したときに性能低下を回避する仕組みが必要である．また，クライアントがサーバにアクセスするタイミングや，ダウンロードする WU のデータサイズを調整し，サーバにおける WU の転送処理を効率化する必要がある．しかし，クライアントは独立してプロジェクトサーバにアクセスしており，クライアント間の連携は行われていない．そのため，WU 配布の効率を最大化し，WU が各クライアントに不足なく行きわたるための仕組みを実現する必要がある．

## 3. 分散協調型スケジューリングによるボランティアコンピューティングの高効率化

本章では，分散協調型スケジューリングによるボランティアコンピューティングの高効率化手法を説明する．まず，本論文で利用する分散協調型スケジューリングについて，その仕組みを説明する．次に，分散協調型スケジューリングによる動的負荷分散およびプロキシダウンロードの 2 つのスケジューリング手法を示し，ボランティアコンピューティングにおける性能低下の問題をどのように解決するか述べる．

### 3.1 分散協調型スケジューリング機構

多数のクライアントを用いる大規模分散計算では，集中型サーバによるクライアントの一括管理は実用的ではない．これに対し，我々は大規模分散計算のための，分散協調型スケジューリング機構を提案している[9]．分散協調型スケジューリング機構では，すべてのクライアントをいくつかの部分集合に分割する．各部分集合内では，1 台のクライアントをマスタとしたマスタ—ワーカ型の構造を持つサブネットワークを形成する．各部分集合では，マスタとなったクライアントによって各クライアントの負荷の比較と，その比較結果に応じて負荷を均一化するための WU の移動が行われる．

クライアントは多数の WU から構成される分散計算ジョブを実行するために，P2P ネットワークを構成する．ここで，各クライアントはローカルタスクを管理するためのローカルスケジューラに加え，分散計算の WU を管理するための分散スケジューラを持つ．分散協調型スケジューリング機構では，P2P の論理リンクで接続されたクライアントどうしの部分集合に対して，スケジューリング処理を適用する．ここで，1 つの部分集合内に含まれるクライアント数を，全クライアント数に比べて非常に少なくすることで，低性能なクライアントにおいてもスケジューリング処理を実施することができるようになる．また，個々のクライアントが持つ分散スケジューラどうしの連携により，すべてのクライアントに対する全体的なスケジューリングを実現する．

### 3.2 クライアントの動的負荷変動に基づく WU の移動機構

BOINC では，サーバによる WU の管理の効率化や分散計算の効率化のために，1 台のクライアントが複数の WU を保持する．しかし，多数の WU を保持したクライアントにおいて負荷変動による計算性能の低下が生じた場合，そのほとんどの WU は BOINC サーバが定めた期限までに計算を完了することができない．その結果，BOINC サーバはそれらの WU を他のクライアントに再割当てしなければならず，ボランティアコンピューティングシステム全体の性能低下を引き起こす．このような問題を解決するためには，クライアントの計算性能の変動に応じた動的負荷分散が必要である．

動的負荷分散を実現するために，提案手法ではクライアント間での WU の移動を行う．まず，各クライアントは自身の処理状況を定期的にモニタリングし，保持している WU の数とそれらの WU の計算が完了するまでの見積り時間を調べる．WU の計算完了時間の見積りには，BOINC が標準で提供している機能を利用する[1),10)]．BOINC は，ボランティアコンピューティングプロジェクトの運営者に対して，各 WU に計算量を設定するように規定している．そのため，各 WU ごとに設定された計算量と各クライアントにおいて計測される処理性能から，各クライアントは WU の計算時間を見積もる．さらに BOINC は，WU の処理時間の推定値と実測値のずれに対して補正を行うため[19)]，より正確な WU の計算完了時間の見積りを行うことができる．次に各クライアントは，隣接する計算資源の WU の計算時間の見積りと，自身の WU の計算時間の見積りを比較し，自身の WU 処理の進み具合を調べる．もし自身の処理能力に比べ保持している WU 数が多すぎるなどの理由により，他のクライアントよりも処理が遅れていると判断されると，余剰 WU をより計算が進んでいる隣接計算資源へ転送する．これにより，あるクライアントが負荷の増加によって処理しきれなくなった WU を，負荷が低い他のクライアントが受け持つことになり，動的負荷分散によりクライアント間での処理の均一化が実現される．また，シャットダウンなどの理由によりクライアントがプロジェクトから離脱する場合，WU を他のクライアントへ転送してから離脱する．これによりクライアントが WU をかかえたままプロジェクトから離脱してしまうことによる WU の消失を防ぐことができる．

### 3.3 プロキシダウンロード

本節では，プロキシダウンロードと名付けた WU ダウンロード機構による，クライアントの WU ダウンロード失敗による計算性能低下の回避，およびサーバにおける WU の転送処理の効率化について述べる．

**図 2** (a) に示すように，BOINC では各クライアントがそれぞれ BOINC サーバから WU のダウンロードを行う．一方，プロキシダウンロード機構を組み込んだ BOINC では，複数のクライアントが必要とする WU を 1 台のクライアントがまとめてダウンロードし，その後負荷分散機構を利用して他のクライアントにダウンロードした WU を転送する．図 2 (b) に，プロキシダウンロードを用いた WU ダウンロードの仕組みを示す．まず，クライアント B は BOINC サーバから WU をダウンロードする際，クライアント A，C，D が計算する分の WU も同時にダウンロードする．その後，クライアント B は分散協調型スケジュー

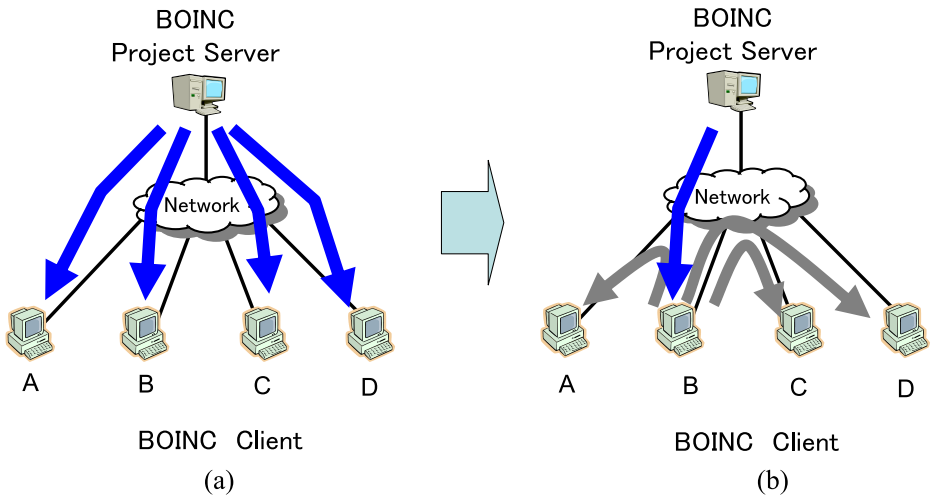


**図 2**　プロキシダウンロード
Fig. 2　Proxy download.

ラを用いてダウンロードした WU を，クライアント A，C，D それぞれに転送する．プロキシダウンロードを用いることで，BOINC サーバが単位時間あたりに配布できる WU 数を変化させずに，BOINC サーバへ単位時間あたりにアクセスするクライアント数を削減することが可能である．これにより，多数のクライアントが同時に BOINC サーバへアクセスすることを防ぎ，BOINC サーバが過負荷状態になることで発生するダウンロード失敗を回避することができる．

次に，BOINC サーバが過負荷時の WU ダウンロードが発生したときの，プロキシダウンロードを用いたクライアントの代替 WU ダウンロードを実現する仕組みについて示す．クライアントが BOINC サーバから WU のダウンロードに失敗した場合，BOINC サーバに対して再度アクセスを行わず，分散協調型スケジューラを介して隣接クライアントが持つ余剰 WU を取得する．これにより，BOINC サーバが一時的に停止していても，各クライアントは継続して WU の処理を行うことができる．その結果，プロキシダウンロードによる，クライアントの利用効率の向上と，ボランティアコンピューティングプロジェクト全体の計算性能の改善を実現する．

プロキシダウンロードを用いることにより，BOINC サーバは 1 回の通信で，複数の WU をクライアントへ転送する．BOINC サーバが 1 回の通信で 1 つの WU しか転送しない場合，WU のデータサイズが小さいとデータ転送に要する時間が短くなるため，通信の遅延や通信プロトコルのヘッダ処理によるオーバヘッド，ディスクアクセスによるオーバヘッドの時間がデータ転送時間に比べて無視できないほど大きくなる．一方，BOINC サーバが 1 度に複数の WU を送信する場合，これらのオーバヘッドを隠蔽し，BOINC サーバにおける処理効率を向上させることができる．さらに，複数の WU を扱うことにより，BOINC サーバの内部処理やデータベースへのアクセスを並列化することが可能であり，BOINC サーバの WU 転送処理を効率的に行うことが可能である．

分散協調型スケジューラによる動的負荷分散およびプロキシダウンロードは，各クライアントでの処理によってのみ実現されている．BOINC サーバの機構は従来のものと同一であり，BOINC によるボランティアコンピューティングシステムと，分散協調型スケジューリング機構は完全に独立して動作している．もし，分散協調型スケジューリング機構による WU 管理が中断されたとしても，BOINC による管理が行われるため，ボランティアコンピューティングシステムが停止するようなことはない．また，分散協調型スケジューリング機構によって生じる処理や通信の負荷は，各クライアントに分散され，BOINC サーバには新たな負荷が生じることはない．

3.1 節で述べたように，分散協調型スケジューリング機構によって各クライアントに生じる負荷の大きさは，クライアントが持つ論理リンクの数に比例する．クライアントあたりの論理リンク数が少なくなるような P2P ネットワークを用いることで，分散協調型スケジューリング機構により発生する処理と通信の負荷はきわめて小さくなる．そのため，提案手法によって各クライアントで発生する負荷が，ボランティアコンピューティングの性能低下を引き起こすことはない．

## 4. 性能評価

本章では，実計算資源を用いた実証実験により，分散協調型負荷分散機構を持たせたボランティアコンピューティング環境の有効性の評価を行う．

提案する分散協調型負荷分散機構を従来のボランティアコンピューティングミドルウエアに機能追加したシステムを構築した．その後，数台規模の計算資源から構成される 2 通りのボランティアコンピューティング環境を構築し，大規模分散データマイニングの性能評価実験を行うことによって，分散協調型負荷分散機構の有効性を明らかにする．

**表 1**　Intrigger プラットフォーム
Table 1　Intrigger platform.

| 拠点名 | CPU 型番 | CPU 動作周波数 | メモリ容量 | ノード数 |
|---|---|---|---|---|
| chiba | Intel Core2Duo 6400 | 2.13 GHz | 4 GB | 57 |
| | Intel Xeon E5410 | 2.33 GHz | 32 GB | 16 |
| kyoto | Intel Core2Duo 6400 | 2.13 GHz | 4 GB | 22 |
| | Intel Xeon E5530 | 2.4 GHz | 24 GB | 7 |
| suzuk | Intel Core2Duo 6400 | 2.13 GHz | 4 GB | 35 |
| okubo | Intel Core2Duo 6400 | 2.13 GHz | 4 GB | 12 |
| mirai | Intel Xeon E5410 | 2.33 GHz | 16 GB | 6 |
| kyushu | Intel Xeon E5410 | 2.33 GHz | 16 GB | 9 |
| kobe | Intel Xeon E5410 | 2.33 GHz | 16 GB | 11 |
| keio | Intel Xeon E5410 | 2.33 GHz | 16 GB | 11 |
| hiro | Intel Xeon E5410 | 2.33 GHz | 16 GB | 11 |
| tohoku | Intel Xeon E5410 | 2.33 GHz | 32 GB | 15 |
| kyutech | Intel Xeon E5410 | 2.33 GHz | 32 GB | 16 |
| tsukuba | Intel Xeon E5410 | 2.33 GHz | 32 GB | 15 |
| huscs | Intel Xeon E5530 | 2.4 GHz | 24 GB | 16 |
| hosei | Intel Xeon E5530 | 2.4 GHz | 24 GB | 18 |
| 計 | | | | 277 |

### 4.1 評価条件

#### 4.1.1 性能評価に用いる分散計算環境

性能評価用のボランティアコンピューティング環境として，特徴の異なる下記の 2 種類の評価環境を用いる．

**Intrigger Platform**　クライアントマシンとして，Intrigger Platform [20] を利用し，広域分散ボランティアコンピューティング環境を構築した．Intrigger Platform は分散計算のテストベッドであり，日本各地に拠点が設けられ，それぞれの拠点にクラスタ並列計算機が設置されている．各拠点間は，学術ネットワーク SINET3 [21] によって相互に接続されている．**表 1** に評価に用いた Intrigger Platform の拠点名と，各拠点におけるノードの諸元性能を示す．1 つのノードにおいて，1 つの BOINC クライアントを起動させ，277 並列の分散計算環境を実現している．

**PlayStation 3**　クライアントマシンとして，家庭用ゲーム機である PLAYSTATION3 (PS3) [22] を利用し，一般家庭の遊休計算資源の活用を想定した分散計算環境を構築した．

近年では，ゲーム機においても一般の計算機と同等の性能向上が進んでおり，なかでも PS3 は高いピーク性能とインターネットに接続する機能をあわせ持っている．しかし，PS3 の利用者は，たとえゲーム中でも，その性能をつねに最大限に使用しているというわけではなく，計算能力の大部分が遊休状態になっている．これは近年の個人用計算機と状況が酷似しており，ゲーム機もボランティアコンピューティングのワーカとして計算資源を提供できると考えられる．ゲーム機もボランティアコンピューティングに参加することができれば，多数の参加者が見込めるため，非常に魅力的な計算資源である．

**表 2**　PlayStation 3 性能諸元
Table 2　PlayStation 3.

| | |
|---|---|
| CPU | Cell Broadband Engine 3.2 GHz |
| SPE 数 | 6 |
| Memory | 256 MB |
| ローカルメモリ | 256 KB/SPE |
| OS | Linux 2.6.27 (Fedora 10) |

PS3 には高性能なプロセッサ Cell Broadband Engine（Cell）[23] が搭載されている．Cell は汎用プロセッサコアである PPE を 1 つと，計算用のコアである SPE を 8 つ持ち，SPE によって並列処理を行うことで高い性能を発揮している．また，計算用のコアである SPE は，アイソレーションモードと呼ばれるセキュリティ強化の機能を持っている．アイソレーションモードで動作する SPE は，PPE や他の SPE からのアクセスが遮断されるため，外部からの読み書きは不可能となる．ボランティアコンピューティングは，クライアントが結果の改竄を行ったり，あるいは計算に必要となる入出力ファイルを盗んだりするといったセキュリティ上の問題点があるが，Cell の持つセキュリティ機能を使用することにより，それらの問題を解決し安全安心なボランティアコンピューティングが可能となる [5]．

評価実験ではクライアントとし 96 台の PS3 を用いる．**表 2** にクライアントとして用いた PS3 の諸元を示す．一般家庭に最も普及している光回線を想定し，各 PS3 は 100 Mbps のイーサネットで接続する．各 PS3 にはオペレーティングシステムとして Fedora 10 をインストールし，ボランティアコンピューティングのソフトウェアを Linux 上で動作させる．

#### 4.1.2 評価アプリケーション

評価アプリケーションとして，データマイニング手法の 1 つである k-means クラスタリング [24] プログラムを用い，実際に科学技術計算データの解析を行い，分散データマイニング環境の有効性を評価する．解析対象データとして非定常流れの数値解析データを利用し，

ロケット噴射ノズル近辺での物理現象の解析を行う[25]．

評価アプリケーションが利用する解析対象データには，数値流体解析によって得られた，計算空間内の格子点上における様々な物理量が数値データとして格納されている．しかし，得られた膨大な解析結果のうち，どの物理量間にデータの相関関係が含まれているか分からないため，任意の物理量データ間での相関関係をデータマイニングによって調べる必要がある．また，最適なクラスタ数についても事前に分からないため，クラスタ数を任意の範囲で変化させ，それぞれのクラスタ数での結果の中から最適な分類がなされたときのクラスタ数を採用する．データマイニングにより適切な分類がなされたかどうかの判断にはシルエット幅[26]を利用する．さらに，k-means クラスタリングの初期値依存性を考慮し，各クラスタの初期値をランダムに変更し，複数回のクラスタリングを実行していくつかのクラスタ分割結果を取得し，それらの分割の中でシルエット幅が最大になるクラスタ分割結果を最終的な解析結果として採用する．

評価アプリケーションは，解析対象物理量の組合せ数（入力データ数），クラスタ数，そして初期クラスタ中心を考慮した繰り返し数（試行回数）の 3 つのパラメータ空間を持つパラメータスウィープ型アプリケーションである．評価アプリケーションではこれら 3 つのパラメータ空間を分割し，個々の WU に対し計算パラメータとして割り当てる．**図 3** に評価アプリケーションの処理構造を示す．評価アプリケーションは，初期クラスタ中心の決定，k-means 処理，シルエット幅計算を 1 つのトランザクション処理とし，入力データ数，試行回数，ならびにクラスタ数によって繰返し数が与えられる 3 重ループ構造を持つ．これにより，WU あたりの計算量は，3 重ループ内で処理されたトランザクション数で表すことができる．WU のデータファイルサイズは，割り当てられた入力データ数に比例し，入力データ 1 つあたり 464 KB である．また，クラスタ数と試行回数には依存せず，ファイルサイズは一定である．

WU あたりのトランザクション数および入力データ数は，特定のアプリケーションに依存しない一般的な特徴として表される．このことから，評価実験の結果よりボランティアコンピューティング環境における一般的な議論が可能である．

#### 4.1.3 評 価 方 法

評価実験では，WU に与える入力データ数，クラスタ数，ならびに試行回数を変化させながら，各評価環境での分散データマイニングを実行し，以下の項目に関して評価を行う．

- **スループット**：ボランティアコンピューティング全体で，単位時間あたりに計算できたトランザクション数．処理できるトランザクション数が多いほどボランティアコンピューティングが高い計算性能を持つことを示し，より大規模なデータマイニングが実現可能となる．
- **サーバ平均 CPU 使用率**：分散計算を行っている最中の BOINC サーバの平均 CPU 利用率．BOINC サーバが管理可能なクライアント数を表す．サーバが高負荷状態になると，WU を割り当てられないクライアントが出現し，ボランティアコンピューティング全体の計算性能が低下する．

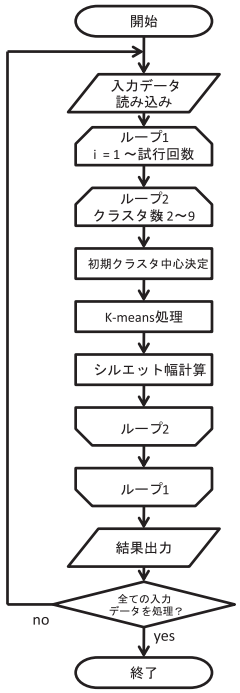


**図 3**　データマイニング処理手順
Fig. 3　Process flow of the datamining application.

サーバに WU を投入した直後では，サーバにおける処理の遅延や，各クライアントがサーバにアクセスするタイミングのばらつきなどにより，不安定要素による性能のばらつきが生じる．そのため，あらかじめサーバに十分な量の WU をあらかじめ投入しておき，ボランティアコンピューティング環境全体の動作が定常状態に落ち着いたときの性能を計測する．

**表 3**　プロジェクトサーバ諸元
Table 3　Server machine.

| | |
|---|---|
| CPU | Intel Core2Duo 2.93 GHz |
| Memory | 4.0 GB |
| OS | Linux 2.6.30 (Fedora 11) |
| Web Server | Apache 2.2.14 |
| BOINC Server | Version 6.11.1 |

評価環境内における計算性能の動的変動の影響を評価するために，BOINC プロジェクトのシミュレーション評価[19] に用いられている，BOINC 稼働率を利用する．BOINC 稼働率は，ボランティアコンピューティングを実行している時間のうち，クライアントマシンが BOINC クライアントを動作させている時間の割合を表す．BOINC クライアントはクライアントマシンが遊休状態のときに動作していることから，BOINC 稼働率はクライアントマシンが遊休状態にある割合も表している．各クライアントマシンは 1 分ごとに BOINC 稼働率に従った動作状態の遷移を行い，BOINC クライアント動作時にはすべての計算能力を BOINC の WU に割り当てるものとする．評価実験では，BOINC 稼働率を BOINC プロジェクトのデフォルト値である 80%とする．分散協調型スケジューリング機構の P2P ネットワークには，リング型のトポロジを用いる．評価環境全体で，1 つのリング型のネットワークを形成する．評価環境として Intrigger Platform を用いるときは，同一拠点内でのリンク接続を優先し，拠点をまたぐリンク接続数を最小化する．

**表 3** にボランティアコンピューティングのプロジェクトサーバの諸元を示す．プロジェクトサーバと Intrigger Platform の間は 1 Gbps のイーサネットで，プロジェクトサーバと PS3 の間は 100 Mbps のイーサネットで接続されている．

### 4.2　最大性能の推定

ボランティアコンピューティングを用いたデータマイニングの性能評価を行う前に，各クライアントマシンを単体で利用したときのデータマイニングの処理性能を評価する．さらに，クライアント単体の処理性能の結果をもとに，評価環境におけるシステムのピーク性能を推定する．各評価環境におけるシステムのピーク性能と，ボランティアコンピューティングを用いてデータマイニングを行ったときの処理性能を比較することで，データマイニング処理の分散化によるオーバヘッドの影響の評価が可能である．

ボランティアコンピューティングを用いずクライアント単体でのアプリケーションを実行したときの平均トランザクション処理時間，およびシステムのピーク性能を，**表 4** に示す．

**表 4**　単体処理性能
Table 4　Single-node performance.

| 評価環境 | Intrigger Platform | PS3 |
|---|---|---|
| トランザクション処理時間［秒］ | 5.4 | 4.3 |
| システムのピーク性能［トランザクション/秒］ | 51.3 | 22.1 |

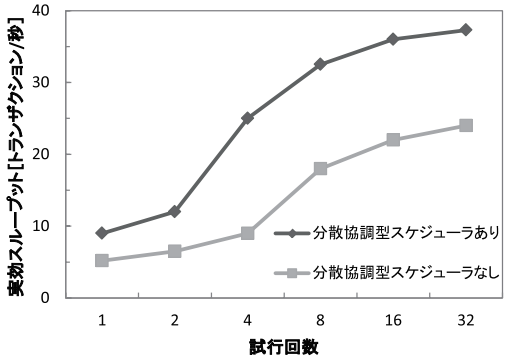


**図 4**　試行回数とスループットの関係（Intrigger Platform）
Fig. 4　Throughput versus number of iterations (Intrigger Platform).

システムのピーク性能は，BOINC による処理の分散化やネットワークのデータ転送オーバヘッドを含まない，評価環境全体が持つ理論的な最大性能である．実際のボランティアコンピューティング環境においては，WU の転送時間や，クライアントの WU 割当て待ち時間などが生じるため，システムのピーク性能より低い値が，実効性能として得られる．

### 4.3　評価結果と考察

#### 4.3.1　スループット

**図 4** と**図 5** に，クラスタ数を 2〜9，入力データ数を 1 とし，試行回数のみを 1〜32 まで変化させたときの，スループットを示す．図 4 は評価環境として Intrigger Platform を用いたときの結果を，図 5 は評価環境として PS3 を用いたときの結果を表している．図の横軸は試行回数を，縦軸はスループットを示しており，分散協調型スケジューラを用いずにデータマイニングを行ったときの結果と，分散協調型スケジューラを導入してデータマイニングを行ったときの結果をそれぞれ示している．

図 4 から分かるように，Intrigger Platform を用いた環境において，分散協調型スケジュー

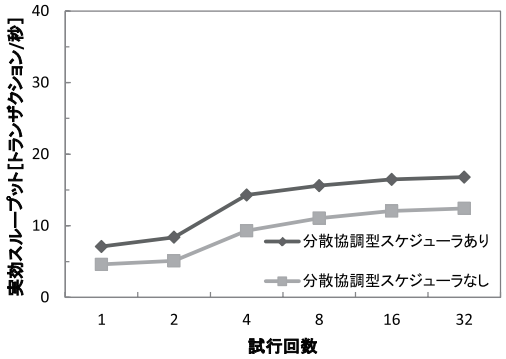


**図 5**　試行回数とスループットの関係（PS3）
Fig. 5　Throughput versus number of iterations (PS3).

ラを用いない場合，試行回数が 1，2，4 回のときには，全体の処理性能は 10［トランザクション/秒］しか達成できていない．同様に図 4 から分かるように，PS3 を用いた環境において分散協調型スケジューラを用いなかった場合，試行回数が 1，2 回のとき，全体の処理性能は 5［トランザクション/秒］しか達成できていない．試行回数が少ない場合，WU あたりの処理時間が短くなるため，クライアントからサーバに対する WU ダウンロード要求が頻繁に発生する．そのため，クライアントからサーバへの過剰なアクセスの集中が発生する．その結果，WU 取得に失敗し，計算を行っていない多数のクライアントが存在するために，全体の処理性能が低くなっている．文献 9) のシミュレーションによる評価において，BOINC が持つ Work Queue 型の構造はアクセス集中による処理性能低下を起こしやすいことが示されており，実計算資源を用いた本評価実験でもクライアントからのアクセス集中による性能低下の問題が確認できる．

また，表 4 の予備実験の結果より，Intrigger Platform は PS3 に比べて約 2 倍のピーク性能を持っている．WU の持つ計算量が同一の条件下では，Intrigger Platform でのクライアントからサーバへの単位時間あたりのアクセス数は，PS3 の場合のおよそ 2 倍となる．このため，試行回数を 4 回とした場合，Intrigger Platform ではサーバへのアクセス集中による性能低下が生じたが，PS3 ではサーバへのアクセス集中とはならず性能低下が回避できている．一方，試行を大きくすると，サーバからの WU 取得に失敗するクライアント数が減少するため，全体の処理性能は試行回数に比例して増加する．しかし，試行回数が十分大きくなると，すべてのクライアントが WU を持つ状態となるため，試行回数を増やしても全体のスループットはわずかしか向上しない．表 4 に示す予備実験の結果との比較によるデータマイニング処理の分散化による実行効率を調べると，分散協調型スケジューラを用いなかった場合のデータマイニングの実行効率は，Intrigger Platform を用いた場合で 70%，PS3 を用いた場合で 58%である．このことから，従来のボランティアコンピューティング環境においては，処理の分散化の効率が低く，多数のクライアントを利用しても高性能な分散計算を実現しにくいことが確認できる．

次に，分散協調型スケジューラの有効性について評価を行う．図 4 と図 5 の評価結果から，分散協調型スケジューラを用いることにより，各条件におけるスループットが平均 50%向上した．特に，評価環境として Intrigger Platform を用い，試行回数を 4 回としたときには，スループットが 2.8 倍になっている．これは，分散協調型スケジューラを用いなかったときサーバが過負荷状態にあったのに対し，分散協調型スケジューラを利用することによってサーバが過負荷状態を回避することができたためである．試行回数が小さい条件での評価は，大規模なボランティアコンピューティング環境における多数のクライアントからのアクセス集中が生じている条件と同等の評価と見なすことができる．このことから，分散協調型スケジューラの導入により，より多くのクライアントを用いた大規模なボランティアコンピューティングにおいて適切な WU 配布が可能であることが示された．

また，試行回数が大きい場合の結果より，分散協調型スケジューラを用いることによってデータマイニングの実行効率が，Intrigger Platform を用いた場合で 91%，PS3 を用いた場合で 95%まで向上していることが分かる．これは，分散協調型スケジューラを用いることで，すべてのクライアントに対し WU を適切に配布できるようになり，WU を持たないクライアントによる計算効率の低下を回避できるためである．

さらに，クライアントのボランティアコンピューティングからの離脱よってそのクライアントに割り当てられていた WU の処理が進まないという問題に対し，動的負荷分散機能による WU のクライアント間の移動によって WU の処理の停止を回避できることも，スループット向上に寄与している．

次に，WU が持つデータ量と計算量の比率を変化させたときの性能を評価する．WU あたりのトランザクション数を一定とし，入力データ数と試行回数を変化させる．トランザクション数は WU の持つ計算量に相当し，入力データ数は WU の通信量に相当する．このことから，本評価では計算量一定の条件下で，計算量に対する通信量の比率を変化させながら性能を評価することになる．**図 6** と**図 7** に，試行回数と入力データ数の組合せ（以下，試行回数/入力データ数）を 1/32 から 32/1 まで変化させたときのスループットを示す．図 6

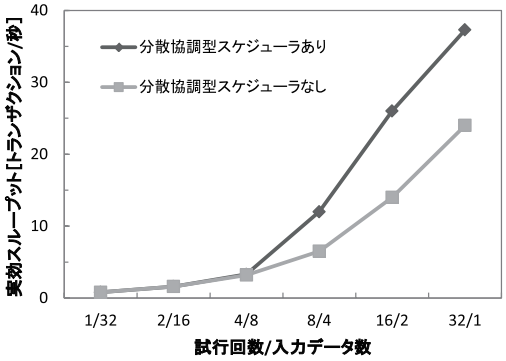


**図 6**　試行回数/入力データ数とスループットの関係（Intrigger Platform）
Fig. 6　Throughput versus number of iterations and input data (Intrigger Platform).

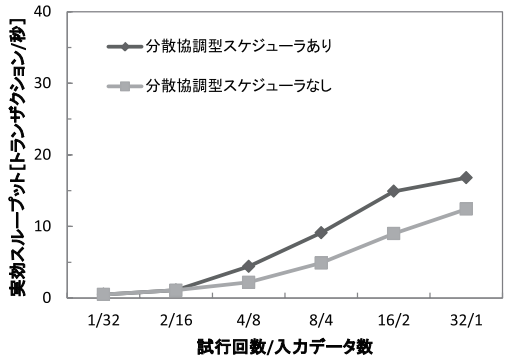


**図 7**　試行回数/入力データ数とスループットの関係（PS3）
Fig. 7　Throughput versus number of iterations and input data (PS3).

は評価環境として Intrigger Platform を用いたときの結果を，図 7 は評価環境として PS3 を用いたときの結果を表している．図の横軸は WU あたりの入力データ数と試行回数を，縦軸はスループットを示しており，分散協調型スケジューラを用いずにデータマイニングを行ったときの結果と，分散協調型スケジューラを導入してデータマイニングを行ったときの結果をそれぞれ示している．

図 6 と図 7 に示した結果より，すべての条件下において，WU の計算量に対する通信量の比率が大きくなるほど，スループットが低くなる傾向が確認できる．これは計算量に対する通信量の比率が大きいほど，サーバからクライアントへの WU のデータ転送時間が処理時間の大部分を占め，クライアントがサーバからの WU ダウンロード待ち状態となり WU の処理が行われないためである．

次に，分散協調型スケジューラを用いた場合の結果と用いなかった場合の結果を比較する．図 6 では試行回数/入力データ数が 8/4 以上の場合で，図 7 では試行回数/入力データ数が 4/8 以上の場合で，分散協調型スケジューラによるスループットの向上が確認できる．この条件下では，WU の持つデータサイズが小さく，またサーバが単位時間あたりに多数の WU を処理する必要がある．このようにサーバが多数の WU を処理する状況下では，プロキシダウンロードによってサーバの WU 処理効率が改善されるため，多数のクライアントに WU を効率的に配布し，システムのスループットが向上する．

一方，図 6 の試行回数/入力データ数が 4/8 以下の場合と，図 7 の試行回数/入力データ数が 2/16 以下の場合では，分散協調型スケジューラによるスループットの向上は確認できない．これらの条件下では，WU の持つデータサイズが非常に大きいため，サーバからクライアントへの WU 転送時間が非常に長い．そのため，サーバにおける I/O アクセスなどのオーバヘッド時間が相対的に短くなり，プロキシダウンロードによるサーバでの WU 処理の最適化が行われない．また，サーバのクライアントへの WU の配布速度は，サーバのネットワークの帯域幅によって律速されており，その結果システム全体では低いスループットしか得られていない．ここで，図 6 と図 7 の結果では，分散協調型スケジューラの効果が現れる試行回数/入力データ数の条件が異なっている．これは，InTrigger Platform を用いた環境の方が，PS3 を用いた環境よりもピーク性能が高く，かつクライアント数が多いため，クライアントからサーバへのアクセス頻度に違いが生じているためである．

図 6 と図 7 より，サーバの単位時間あたりに処理する WU 数が多いほど，提案する分散協調型スケジューラによる性能向上が大きい．一方で，WU あたりのデータサイズが大きいと，サーバのネットワーク帯域幅がボトルネックとなり，サーバが単位時間あたりに処理可能な WU 数が大きく制限される．ネットワーク帯域幅がボトルネックになるという問題は，BOINC が持つ Work Queue 型の基本機構の本質的問題であり，分散協調型スケジューラのみを用いた問題解決は難しい．この問題の解決法としては，WU を計算に関する基本情報部分と，計算に使用するデータファイルに分割し，データファイルを外部の分散ファイルサーバなどに格納する手法が考えられる[27]．サーバは，基本情報のみを保持したデータサイズの小さな WU を処理すればよいため，ネットワーク帯域幅によるデータ転送の影響を削減でき，単位時間あたりに処理可能な WU 数が増加する．したがって，サーバにおける

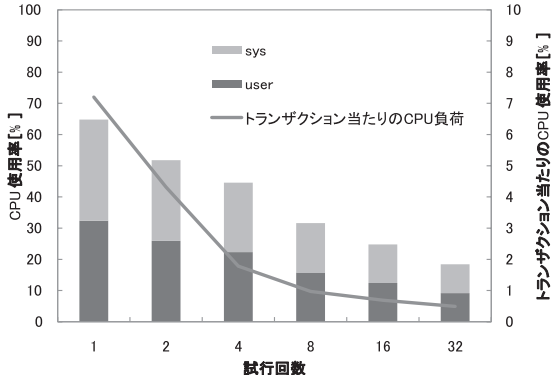


**図 8**　CPU 利用率（Intrigger Platform，分散協調型スケジューラあり）
Fig. 8　CPU load (Intrigger Platform, with distributed and cooperative scheduler).

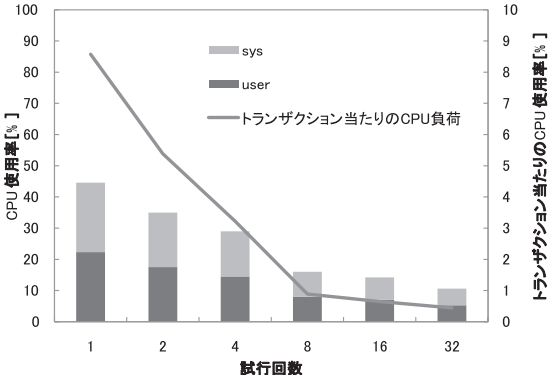


**図 9**　CPU 利用率（Intrigger Platform，分散協調型スケジューラなし）
Fig. 9　CPU load (Intrigger Platform, without distributed and cooperative scheduler).

単位時間あたりに処理可能な WU 数が多いほど，提案する分散協調型スケジューラによる性能改善の効果は高くなる．そのため，巨大なデータファイルを扱う分散環境においても，分散ファイルサーバなどとの連携することで，提案手法を有効に機能させることが期待できる．

### 4.3.2　サーバ負荷

次に，クライアントからのアクセスによってサーバに生じる負荷の大きさを評価するために，各条件におけるサーバの平均 CPU 利用率を評価する．評価環境として Intrigger Platform を用い，分散協調型スケジューラを利用したときのサーバの CPU 利用率を**図 8** に，分散協調型スケジューラを利用しなかったときのサーバの CPU 利用率を**図 9** に示す．同様に，評価環境として PS3 を用い，分散協調型スケジューラを利用したときのサーバの CPU 利用率を**図 10** に，分散協調型スケジューラを利用しなかったときのサーバの CPU 利用率を**図 11** に示す．図の横軸は試行回数を示し，左の縦軸はサーバの CPU 利用率を示している．図の右の縦軸は，トランザクションあたりの CPU 利用率を示しており，サーバの CPU 利用率をシステム全体のスループットで除算した値である．トランザクションあたりの CPU 利用率は，サーバの WU の管理処理がどれだけ効率的に行われているかを表しており，値が小さいほどサーバにおける処理が効率的に行われていることを示している．

すべての実験条件に共通する傾向として，試行回数が少ないほどサーバに生じる負荷が大きく，試行回数を増やすことでサーバの負荷が減少することが確認できる．サーバに生じる

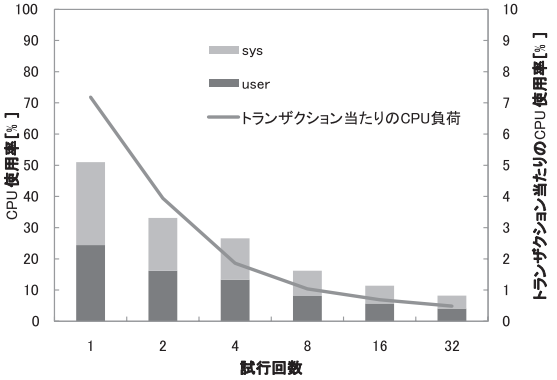


**図 10**　CPU 利用率（PS3，分散協調型スケジューラあり）
Fig. 10　CPU load (PS3, with distributed and cooperative scheduler).

負荷は単位時間あたりにサーバにアクセスするクライアントの台数に依存するため，試行回数が小さい場合ほど，短時間の内に多くのクライアントが同時にサーバにアクセスするため，サーバはこれらの通信への応答が間に合わなくなる．このような状態を，以下，サーバの過負荷状態と呼ぶ．

分散協調型スケジューラを用いない場合，サーバが過負荷状態にあるときの CPU 利用率

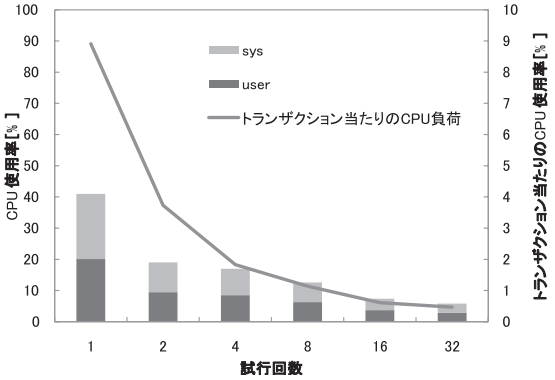


**図 11**　CPU 利用率（PS3，分散協調型スケジューラなし）
Fig. 11　CPU load (PS3, without distributed and cooperative scheduler).

は図 9 で 40〜30%，図 11 で 40〜20%であることが分かる．CPU 利用率が 50%未満にもかかわらず，サーバは過負荷状態となり，WU の配布が適切に行われていない．一方，分散協調型スケジューラを用いた場合には，システム全体のスループット向上に起因する CPU 利用率の増加が発生しているにもかかわらず，図 8 および図 10 においてサーバの過負荷状態は確認できない．ここで，同一評価環境でのトランザクションあたりの CPU 利用率を比較すると，試行回数が 1，2，4 回の範囲において，分散協調型スケジューラを用いたときの CPU 利用率は，分散協調型スケジューラを用いなかった場合に比べて約 2%低くなっている．これは，プロキシダウンロードの WU の一括ダウンロードによって，サーバが個別に WU を処理することによって発生していたオーバヘッドが削減されたためであり，サーバにおける WU 処理の高効率化が確認できる．以上より，サーバの処理能力には限界があるが，分散協調型スケジューラによるサーバでの処理の高効率化により，サーバが一定時間あたりに応答可能なクライアントからの WU 要求の改善が実現できることが示された．

また，すべての実験条件において，サーバに生じる CPU 負荷の半分がディスクアクセスやネットワークアクセスなどの I/O 処理に費やされていることが確認できる．大規模のボランティアコンピューティングを効率的に行うためには，クライアントの台数に応じた性能のサーバが必要とされるが，CPU の処理性能だけではなく，ディスクやネットワークのアクセス性能まで考慮したサーバシステムの構築が重要であることが明らかになった．

## 5. おわりに

本論文では，高性能なボランティアコンピューティングを実現するための分散協調型スケジューリング機構を提案した．提案手法は，ボランティアコンピューティング環境に対し，動的負荷分散機能とプロキシダウンロード機能を提供する．2 種類の分散計算環境を用い，提案する動的負荷分散機構を組み込んだボランティアコンピューティング環境を構築し，実科学技術データを用いた分散データマイニングによる性能評価を行った．評価結果より，提案手法によって実行効率が 90%を超える高い分散データマイニングの実行効率を実現できることを示した．

さらに大規模なボランティアコンピューティング環境では，WU を転送する際のネットワーク性能がボトルネックとなり，計算効率が低下するという問題がある．今後は，計算量とデータサイズが異なる複数の評価用データを用いてネットワークを介したデータ転送の詳細な評価を行い，高効率な分散計算の実現に向けたデータ転送のオーバヘッド削減の検討を行う必要がある．さらに，計算資源を性能の異なるネットワーク環境に分散配置した実証実験環境を構築し，ネットワーク性能が大きく影響する大規模環境下での性能評価を行う．また，各クライアントにおける WU の移動を行うための判断基準の詳細化を行う．現在は，各クライアントは 1 つのボランティアコンピューティングプロジェクトに参加することを前提としているが，今後は複数プロジェクトに参加したときの WU 転送基準について検討を行う．

**謝辞**　多くの貴重なコメントをいただいた査読者の方々に深く感謝いたします．本研究を行うにあたり多くのご支援をいただいた東北大学流体科学研究所の大林茂教授に深く感謝いたします．また，実験に用いた InTrigger プラットフォームを構成，管理されている皆様に感謝いたします．本研究の一部は，文部科学省科研費特定領域研究（18049003），および総務省特定領域重点型研究開発（061102002）の支援を受けて行われた．

## 参考文献

1) Anderson, D.P., Korpela, E. and Walton, R.: High-Performance Task Distribution for Volunteer Computing, *1st IEEE International Conference on e-Science and Grid Technologies*, pp.196–203 (2005).
2) Anderson, D.P., Cobb, J., Korpela, E., Lebofsky, M. and Werthimer, D.: SETI@home: An experiment in Public-Resource Computing, *Comm. ACM*, Vol.45, No.11, pp.56–61 (2002).

3) Folding@home distributed computing. http://folding.stanford.edu/
4) Wang, H., Takizawa, H. and Kobayashi, H.: Performance Evaluation of K-Means Clustering on the Cell Processor, *Proc. High Performance Computing Symposium 2007*, pp.161–168 (2007).
5) Wang, H., Takizawa, H. and Kobayashi, H.: A Performance Study of Secure Data Mining on the Cell Processor, *CCGRID '08: Proc. 2008 8th IEEE International Symposium on Cluster Computing and the Grid*, Washington, DC, USA, IEEE Computer Society, pp.633–638 (2008).
6) Casanova, H., Legrand, A., Zagorodnov, D. and Berman, F.: Heuristics for Scheduling Parameter Sweep Applications in Grid Environments, *Proc. 9th Heterogeneous Computing Workshop* (*HCW 2000*), pp.349–363 (2000).
7) Casanova, H., Hayes, J. and Yang, Y.: Algorithms and Software to Schedule and Deploy Independent Tasks in Grid Environments, *Proc. Workshop Distributed Computing, Metacomputing, and Resource Globalization* (2002).
8) Berman, F., Wolski, R., Casanova, H., et al.: Adaptive Computing on the Grid Using AppLeS, *IEEE Trans. Parallel and Distributed Systems*, Vol.14, No.4, pp.369–382 (2003).
9) 村田善智，稲葉　勉，滝沢寛之，小林広明：大規模計算環境における分散協調型負荷分散手法，情報処理学会論文誌特集号 “新しいパラダイムの中での分散システム/インターネット運用・管理”，Vol.3, No.49, pp.1214–1228 (2008).
10) Anderson, D.P.: BOINC: A System for Public-Resource Computing and Storage, *5th IEEE/ACM International Workshop on Grid Computing*, pp.4–10 (2004).
11) distributed.net. http://www.distributed.net/
12) Einstein@Home. http://einstein.phys.uwm.edu/
13) Predictor@Home. http://predictor.scripps.edu/
14) SZTAKI Desktop Grid. http://szdg.lpds.sztaki.hu/szdg/
15) ClimatePrediction.net. http://climateprediction.net/
16) LHC@home. http://lhcathome.cern.ch/lhcathome/
17) TANPAKU. http://issofty17.is.noda.tus.ac.jp/index.php
18) PS3GRID. http://www.ps3grid.net/PS3GRID/
19) “SourceCode–BOINC–Trac”. http://boinc.berkeley.edu/trac/wiki/SourceCode
20) Saito, H., Kamoshida, Y., Sawai, S., Hironaka, K., Takahashi, K., Sekiya, T., Dun, N., Shibata, T., Yokoyama, D. and Taura, K.: InTrigger: A Multi-Site Distributed Computing Environment Supporting Flexible Configuration Changes, *Proc. Summer United Workshops on Parallel, Distributed and Cooprative Processing* (*SWoPP2007*), pp.237–242 (2007).
21) NationalInstituteofInformatics (NII): Next-generatiion Science Information NETwork 3: SINET3 pages. http://www.sinet.ad.jp/?set_language=en
22) Sony Computer Entertainment Inc.: “PLAYSTATION3 OFFICIAL SITE”, http://www.jp.playstation.com/ps3/
23) Kahle, J.A., Day, M.N., Hofstee, H.P., Johns, C.R., Maeurer, T.R. and Shippy, D.: Introduction to the Cell multiprocessor, *IBM Journal of Research and Development*, Vol.49, No.4.5, pp.589–604 (2005).
24) MACQUEEN, J.: Some methods for classification and analysis of multivariate observations, *Proc. 5th Berkeley symposium on mathematical statistics and probability, 1967*, Vol.1, pp.281–297 (1967).
25) Obayashi, S., Jeong, S. and Shibasaki, T.: Construction of a Data Mining Method for Unsteady Flow Field, 年次大会講演論文集：JSME annual meeting, Vol.2007, No.6, pp.163–164 (2007).
26) Rousseeuw, P.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, *Journal of Computational and Applied Mathematics*, Vol.20, pp.53–65 (1987).
27) Ranganathan, K. and Foster, I.: Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids, *Journal of Grid Computing*, Vol.1, pp.53–62.

(平成 22 年 5 月 31 日受付)
(平成 22 年 11 月 5 日採録)

**村田　善智**

平成 20 年東北大学大学院情報科学研究科博士後期課程修了．博士（情報科学）．平成 20 年東北大学サイバーサイエンスセンター産学連携研究員．Grid コンピューティング，ボランティアコンピューティング，P2P コンピューティングおよびベクトルスーパコンピュータの Grid 連携に関する研究に従事．IEEE CS 会員．

**石杜　佑記**

平成 20 年東北大学工学部機械知能・航空工学科卒業．現在，同大学大学院情報科学研究科博士前期課程に在学し，ボランティアコンピューティングに関する研究に従事．

**滝沢　寛之**（正会員）

平成 11 年東北大学大学院情報科学研究科博士課程修了．博士（情報科学）．同年新潟大学総合情報処理センター助手，平成 15 年東北大学情報シナジーセンター助手，平成 16 年同大学大学院情報科学研究科講師を経て，平成 21 年より同研究科准教授．高性能計算システム，コンピュータアーキテクチャとその応用に関する研究に従事．平成 16 年 ISPA04 最優秀論文賞，平成 18 年船井情報科学奨励賞，平成 20 年情報処理学会東北支部野口研究奨励賞，平成 21 年石田記念財団研究奨励賞受賞．電子情報通信学会，IEEE CS 各会員．

**小林　広明**（正会員）

昭和 63 年東北大学大学院博士課程修了．同年東北大学助手．平成 3 年東北大学講師．平成 5 年東北大学助教授．平成 13 年東北大学教授（平成 20 年 4 月よりサイバーサイエンスセンター長兼任）．平成 18 年より NII 客員教授併任．コンピュータアーキテクチャ，並列処理システムとその応用に関する研究に従事．工学博士．IEEE Senior Member，ACM，電子情報通信学会各会員．