Vol.2014-HPC-144 No.7
2014/5/26

Webベースボランティアコンピューティングのためのブラウザ間P2P通信機能

工原 誠[1]　渡邊 寛[1,a)]　福士 将[2]　天野 憲樹[3]　野上 保之[1]

概要：本研究グループでは，Web ブラウザを用いて簡単に参加できる，Web ベースのボランティアコンピューティング (Web ベース VC) を提案している．Web ベース VC の問題点として，全ての処理を Web ブラウザ上で行うため，近年 VC で注目されつつあるユーザ間ファイルコピー機能や同期機能などの実装に必要となる，ユーザ間の直接通信が容易でないという点が挙げられる．そこで本研究では，Web ブラウザ間の P2P 通信を実現する技術として策定が進められている Web Real-Time Communication (WebRTC) とそのライブラリである PeerJS を用いて，Web ベース VC におけるユーザ間の直接通信を実現する．また，通信機能の利用例として，巨大なファイルを配布する際にユーザ間での P2P ファイルコピーを行うことで，全ユーザへの配布完了に要する時間を従来の半分程度まで短縮できることを示す．

キーワード：Web，　WebRTC, Peer-to-Peer, 並列分散処理, デスクトップグリッド

Peer-to-Peer Communication Function among Web Browsers for Web-based Volunteer Computing

Makoto KUHARA[1]　Kan WATANABE[1,a)]　Masaru FUKUSHI[2]　Noriki AMANO[3]
Yasuyuki NOGAMI[1]

Abstract: As an accessible and high-performance computing environment, we have studied Web-based Volunteer Computing (Web-based VC) in which users can join with just accessing a specified URL with Web browsers. Because Web-based VC handles all processes on Web browsers, some of advanced functions such as file sharing and direct communications among users have been difficult to implement. To realize direct communications among users on Web-based VC, this paper proposes a P2P communication function among Web browsers by using Web Real-Time Communication (WebRTC) API and its library, PeerJS. We demonstrated the P2P communication function by implementing file sharing on Web browsers, which cut the total time of distribution in half by duplicating large files among users.

Keywords: Web，HTML5, LLVM, Parallel Computing, Desktop Grids

1. まえがき

ボランティアコンピューティング (VC) は，家庭用のパーソナルコンピュータ (PC) で普段使用されていない余剰計算資源をインターネットを通じて提供してもらい，大規模な並列計算システムを構築する手法である．SETI@home[1] などに代表されるように，VC は，数百万人の参加者を集めることでスーパーコンピュータ並の高い性能が実現可能である．一定以上の時間がかかる大規模な計算に対しては，VC を用いる方が AmazonEC2 クラウドよりも低コストであるという調査結果 [12] が報告されており，安価で高性能な計算プラットフォームとしての VC が近年注目されている．

本研究グループでは，PC の扱いに不慣れなユーザでも，Web ブラウザで指定 URL にアクセスするだけで手軽に参

[1] 岡山大学　大学院自然科学研究科
[2] 山口大学　大学院理工学研究科
[3] 埼玉大学　基盤教育研究センター
[a)] kan.watanabe@okayama-u.ac.jp

加できるような VC(Web ベース VC) を提案している [14]. Web ベース VC は，BOINC[2] のようなミドルウェアを用いた VC と比較して，クライアントソフトウェアのダウンロード・インストール等を必要としないため, ユーザが任意の計算プロジェクトへ容易に参加することが可能となり, より多くの参加者を集めることが期待される.

Web ベース VC の問題点として，参加者が行う全ての処理を Web ブラウザ上で実現する必要があるため, 近年 VC で注目されつつあるユーザ間ファイルコピー機能や同期機能などの実装に必要となる，ユーザ間の直接通信が容易でないという点が挙げられる．これに対し，近年，HTML5 による Web 規格の標準化に伴い Web ブラウザの高機能化が進んでおり，旧来は実現できなかったような Web サービスが盛んに開発・提供されている.

本研究では，Web ベース VC 上でユーザ間の直接通信を実現するために，HTML5 の新機能の 1 つである Web Real-Time Communication (WebRTC) に着目した．WebRTC は，Web ブラウザ間で P2P 通信を行う技術であり，その JavaScript ライブラリである PeerJS[7] が利用可能である．そこで本研究では，PeerJS を使用して，Web ベース VC におけるユーザ間 P2P 通信機能を実装した．また，通信機能の利用例として，巨大なファイルを P2P 通信によってユーザ同士の間でコピーする機能を実装し，全ユーザにファイルのコピーが配布されるまでの時間を計測する性能評価実験を行った.

以下では，2 章において Web ベース VC の概要を述べ，3 章では関連研究および WebRTC 等のブラウザ間 P2P 通信技術について説明する．4 章では，P2P 通信機能の利用例として，ブラウザ間 P2P 通信を用いたファイル配布機能を実装する．5 章では，ファイルコピー機能の性能評価実験を行い，6 章にて結論と今後の課題を述べる.

2. Web ベースボランティアコンピューティング

2.1 VC への参加方法

Web ベースボランティアコンピューティング（Web ベース VC）とは, 本研究グループが提案する, ブラウザを介して計算プロジェクトへ参加する VC システムである [14].

現在主流となっているミドルウェアである BOINC を用いた VC では，参加者が計算プロジェクトに参加する際, 以下の手順で作業を行う必要がある.

(1) BOINC クライアントソフトのダウンロード
(2) クライアントソフトのインストールと PC の再起動
(3) クライアントソフトの起動
(4) 参加する VC プロジェクトの選択
(5) メールアドレスによるユーザ登録

一方，Web ベース VC では，参加者は以下の手順のみで

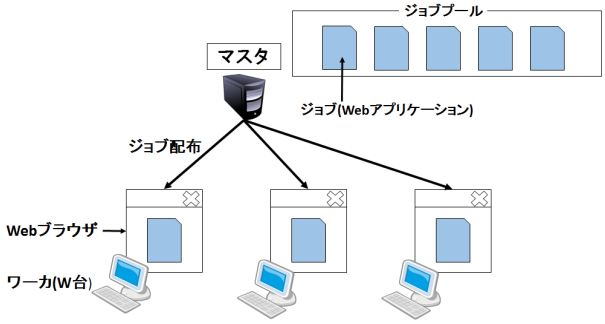


図 1 Web ベース VC の計算モデル

容易に VC へ参加が可能なため，BOINC を利用した VC システムよりも多くの参加者が集まる事が期待される.

(1) Web ブラウザの起動（(2) と同時に実行可能）
(2) 参加する VC プロジェクトの指定 URL へのアクセス

2.2 計算モデル

Web ベース VC における計算モデルは, BOINC を利用した既存 VC プラットフォームと同様, システム全体の管理を行う管理ノード (マスタ) と, 参加者の PC 等 (ワーカ) を構成要素とするマスタ・ワーカモデルである．本モデルの概要を図 1 に示す．本モデルにおける計算は次の手順で行われる.

- マスタは巨大な計算プロジェクトを独立した N 個の計算問題 (ジョブ) に分割し，ワーカからのジョブ要求に応じて個々のワーカにジョブを配布する.
- ジョブを配布されたワーカはこれを実行し，生成された計算結果 (リザルト) をマスタへ返却する.
- 計算プロジェクトは，N 個のジョブ全てが終了すれば完了となる.

2.3 計算の高信頼化

VC における参加者は, VC のプロジェクトに興味を持ち善意を持って参加する者ばかりとは限らず, 悪戯やプロジェクトの妨害等を目的とした，悪意ある参加者 (妨害者) が少なからず存在する [11]. このため BOINC では, 1 つのジョブに対して一定個数のリザルトを集め多数決を行うことで誤ったリザルトを排除し，システム全体の高信頼化を図っている．また BOINC では, 参加者に対して「メールアドレスによるユーザ（ワーカ ID）登録」等の手間を課すことで, 妨害者の攻撃をある程度抑制している.

一方，Web ベース VC においては, ユーザ登録等のステップを排除することで，Web ブラウザによるアクセスだけで手軽に参加できることを実現している．このため単純な多数決では，何度もワーカ ID を変えて誤ったリザルトを返すような，悪意ある妨害者を排除することが困難であった．これに対し本研究グループでは，このような攻撃に対して有効な高信頼化手法として，信頼度に基づく多数

決法 [16] を提案している．同手法では, ワーカそれぞれに対して過去の計算実績に基づいて信頼度を計算し，信頼度を重みとした多数決を行う．これにより，再参加を繰り返すような妨害者は他のワーカより信頼度が低くなるため，計算結果の信頼性に与える影響を低減させることが可能となる．

2.4 LLVM による計算問題の Web アプリケーション化

Web ベース VC では，ジョブの実行を含むワーカ側の処理全てが Web ブラウザ上で行われる．しかし，既存の VC システムで扱われていたジョブは C/C++で記述された科学技術計算問題などが多く，これらを直接 Web ブラウザ上で実行することは困難である．また，JavaScript 等で記述したジョブでは, 機能が制限されたり，十分な計算性能が出なかったりする場合が多いといった問題がある．

これに対し本研究グループでは，LLVM[15] を用いることで既存の C/C++コードを Web ベース VC のジョブへ変換する，Web アプリケーション化手法を提案している [14]．現在までに，姫野ベンチマークを対象とした実験により，Chrome ブラウザ上で LLVM ビットコードを直接実行する PNaCl[4] や JavaScript に対して静的な型情報を付加する asm.js[3] を用いることで，ネィティブプログラムとほぼ同等の性能を維持しつつ Web アプリケーション化できることが分かっている．

3. 関連研究

3.1 ワーカ間 P2P 通信による動的負荷分散

[13] では，VC のような大規模並列計算システムに対して P2P 通信を用いた動的な負荷分散機能を実装することにより, システム全体の性能が最大で約 3 倍向上することが示されている．この負荷分散機能は, 計算プロジェクトに参加しているワーカ同士が図 2 のような P2P 通信によるオーバーレイネットワークを形成し, 隣接するワーカ同士の負荷を数値化したものが一定範囲内に納まるように，互いのジョブ量を調節するものである．このような機能は，Web ベース VC においても効果的であると考えられるが，ブラウザ上で全ての処理を行う Web ベース VC では実装が困難であった．

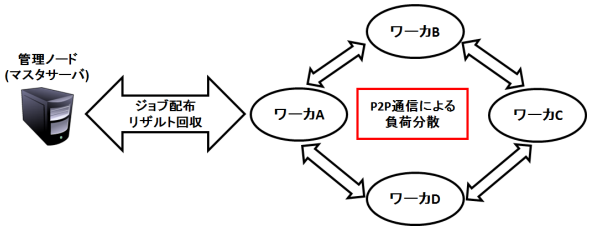


図 2 ワーカ間 P2P 通信による動的負荷分散機能の概要

3.2 Web ブラウザ間通信機能

Web は, 使用の容易性・高い汎用性から様々な用途で用いられており, 現在も多様なサービスが Web アプリケーション (HTML や HTTP/HTTPS などを用いたネットワーク通信を用いて Web ブラウザ上で動作するアプリケーション) として提供されている．Web ブラウザ間通信機能は，近年の Web 技術の進歩により実装が可能となった，HTML5 により実現される最新の Web ブラウザ機能の一つである．本機能を利用することにより, 近年 VC で注目されつつあるユーザ間ファイルコピー機能や同期機能, 3.1 節で述べた P2P による負荷分散機能の実装が可能となる．

以下では, 本研究で用いた HTML5 の技術やライブラリを紹介する．

(1) WebSocket[6]：
WebSocket は，インターネット上で双方向通信を行うためのプロトコルであり，当初は HTML5 の仕様の一部として，現在は HTML5 と独立して策定が進められている．従来，Web ページのリアルタイム表示に利用されていた Ajax や Comet のデメリット部分である，通信のためのリソース消費量を改善し，効率的な双方向通信を可能にしている．また，サーバ・クライアント間で一度接続を確立させれば, 明示的に切断しない限り通信手順を意識せずにデータの送受信が可能である．

(2) WebRTC (Web Real-Time Communications)[5]：
WebRTC は，Web ブラウザに対してプラグインを追加することなく，WebSocket による双方向リアルタイム通信を実現する API である．WebRTC を用いた Web ブラウザ間通信は，2011 年に Google によって提唱され, W3C および IETF にて API・プロトコルレベルでそれぞれ標準化が進められている．

(3) PeerJS[7]：
PeerJS は，WebRTC を用いたブラウザ間 P2P 通信機能をより容易に実装するための JavaScript ライブラリである．通信はハイブリッド P2P 通信方式で行われ，クライアントは仲介サーバ (PeerServer) を経由して PeerID(クライアントを識別するための英数字 13-16 桁の ID) を取得する．通信相手の PeerID を知ることで，Web ブラウザ上のみで P2P 通信が可能となる．

4. Web ブラウザ間 P2P 通信を用いたファイル配布機能の実装

4.1 概要

本研究では，Web ブラウザ間 P2P 通信機能の利用例として，ブラウザ間 P2P 通信によってワーカ同士でファイルをコピーする機能を実装することで, マスタがワーカ全員に直接ファイルを配布する場合と比べて, 配布完了までに要する時間を短縮できる事を示す．

性能比較のため，マスタがワーカ全員に対して直接ファイルを配布するマスタ・ワーカ型ファイル配布機能と，ブラウザ間 P2P 通信によってワーカ間でファイルをコピーする P2P 型ファイル配布機能の二種類を実装する．両者の共通部分について 4.2.1 節で述べ，次に各機能の詳細な実装についてそれぞれ 4.2.2 節と 4.2.3 節で述べる．

P2P 型ファイル配布機能では，マスタ・ワーカ型ファイル配布機能と比較して，ファイル配布完了までに要する時間が短縮されることが期待される．そこで 4.3 節では，ファイル配布完了に要する時間を最も短くするような配布方法について議論する．

4.2 ファイル配布機能の実装

4.2.1 共通部分の実装

本研究で実装するマスタ・ワーカ型ファイル配布機能と P2P 型ファイル配布機能における，両者の共通部分の実装について述べる．配布するファイルは，初期状態においてマスタが保持しているものとし，このファイルのコピーを全ワーカに配布する．

マスタは Node.js[8] を用いて JavaScript のみで実装した．ワーカとの通信は WebSocket によるソケット通信で行う．マスタは，以下の 2 つの情報を変数として保持する．

(1) workerList
参加ワーカの情報 (ID, ファイル配布状況等) を記録する．ワーカからのファイル要求や，ファイル受信完了通知を受け取った際に情報が更新される．

(2) finFileSend
ファイルのコピーが配布済みとなったワーカの台数を保持する．ワーカからのファイル受信完了通知を受け取った際に更新される．

ワーカは HTML5 と JavaScript を用いて実装した．図 3 のように, Web ブラウザにおけるタブ 1 つがワーカ 1 台に相当し, 異なるタブで同じ URL を開いた場合や, 既にワーカを起動しているタブを更新した場合は, それぞれ異なるワーカとして ID で識別される．

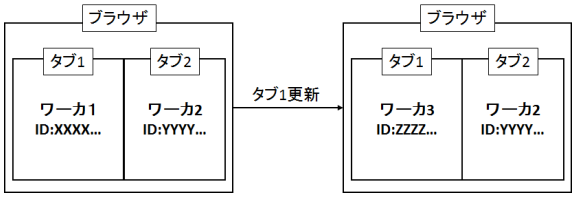


図 3 Web ベース VC におけるワーカ

ワーカの動作は次の通りである．まず，Web ブラウザで特定の URL を開くことでワーカが起動し, WebSocket を用いてマスタとの通信が開始される．この際，マスタに対して，ワーカ ID と共にファイル要求が送信される．マスタや他のワーカから配布されたファイルは，FileAPI によってブラウザ内データベースへ保存される．保存が完了すると，ワーカはマスタに対してファイル受信完了通知を送信する．

4.2.2 マスタ・ワーカ型ファイル配布機能

図 4 に, マスタ・ワーカ型ファイル配布機能における処理の流れを示す．起動したワーカは，マスタに対し，ユニークな ID と共にファイル要求を送信する．マスタは，受け取った ID が workerList に登録されているかどうかを確認し, 登録されていない場合は新規登録を行う（ID 格納）．

マスタ・ワーカ型ファイル配布機能では，マスタが直接，ファイル要求を送信したワーカに対してファイルを送信する．ワーカは，受け取ったファイルを FileAPI によってブラウザ内データベースへ保存すると共に，ファイル受信完了通知を送信する．この通知を受け取ったマスタは，finFileSend の値をインクリメントすると共に，workerList を更新する．

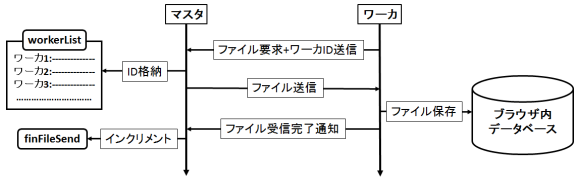


図 4 マスタ・ワーカ型ファイル配布機能の流れ

4.2.3 P2P 型ファイル配布機能

P2P 型ファイル配布機能では，マスタ・ワーカ型ファイル配布機能とは異なり，ファイルの受信が完了したワーカから，ファイルを未受信のワーカへ，ワーカ間でファイルのコピーが行われる．これにより, 従来マスタの負荷となっていたファイル配布によるディスク I/O を抑え，マスタの通信帯域等の資源消費を低減することが可能である．負荷が軽減される分，マスタはジョブの進捗管理等に自身の能力を使用出来るためシステムの大規模化が可能であり，VC システム全体の性能向上につながることが期待できる．

ワーカ間のファイルコピーを実現するため，本研究ではワーカとマスタそれぞれに対して次の機能を追加した．

(1) ワーカ：
P2P 型ファイル配布機能では，PeerJS を用いたワーカ間 P2P 通信のために, 各ワーカが PeerID を持つ必要がある．よって各ワーカは，起動した直後に PeerServer へアクセスして PeerID を取得し，これをファイル要求と共にマスタへ送信する．ワーカは，その役割に応じて中間ワーカと子ワーカの 2 種に分かれる．中間ワーカは, マスタから直接ファイルを受け取るワーカであり, 子ワーカからの要求に応じて P2P 通信を用いてファイルをコピーする．子ワーカは, マスタから受け取った中間ワーカの PeerID を用いて中間ワーカにファイル要求を行い，ファイルのコピーを受け取る．

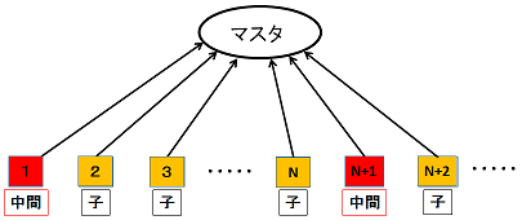


図 5 中間ワーカ判定方法

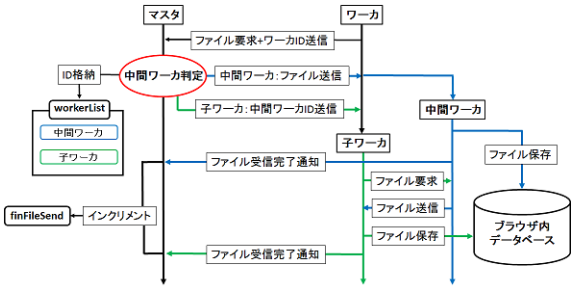


図 6 P2P 型ファイル配布機能の流れ

(2) マスタ：
マスタでは，PeerJS ライブラリを利用したブラウザ間 P2P 通信を行うため，仲介サーバである PeerServer をマスタ内で起動させておく．また，各ワーカを中間ワーカ・子ワーカのいずれかに分類する中間ワーカ判定機能を持つ．

マスタにおける中間ワーカ判定には様々な方法が考えられるが，本実装では，ワーカがアクセスしてきた順番によって中間ワーカかどうかを判定することとした．中間ワーカ判定は図 5 のように行われる．中間ワーカ 1 台につき子ワーカ N-1 台までファイルを配布することとする．まず, 1 台目にアクセスしてきたワーカを中間ワーカと判定し, マスタはそのワーカの PeerID を中間ワーカ ID として保持した後, ファイルを配布する．2 台目から N 台目にアクセスしてきたワーカは子ワーカと判定し, マスタは子ワーカに対して中間ワーカ ID を通知する．N+1 台目以降も同様である．

このような中間ワーカ判定方法を用いると，P2P 型ファイル配布機能における処理の流れは図 6 のようになる．まず，ワーカが起動してマスタにファイル要求を送ると，1 台目のワーカは中間ワーカと判定される．この中間ワーカは，マスタからファイルを受け取り保存した後，子ワーカからのファイル要求を待つ．2 台目から N 台目までのワーカが起動すると，マスタはこれらのワーカを子ワーカと判定するため，各子ワーカは中間ワーカの PeerID を受け取る．各子ワーカは，この PeerID を用いて中間ワーカへファイル要求を行い，ファイルを受け取って保存した後，マスタに対してファイル受信完了通知を送付する．

4.3 中間ワーカ台数の最適値

P2P 型ファイル配布機能では，中間ワーカから子ワーカへファイルをコピーすることで，マスタの負荷を軽減すると共に，ファイル配布完了までに要する時間を短縮することができる．ただし，ワーカ間の通信帯域幅は有限であるため，中間ワーカ台数が少なすぎる場合は中間ワーカ・子ワーカ間のコピーに時間がかかってしまい，逆に中間ワーカが多すぎる場合は，マスタ・中間ワーカ間のファイル配布に時間がかかってしまう．すなわち，P2P 型ファイル配布機能におけるファイル配布に要する時間は，中間ワーカ台数に依存することが分かる．

以下に示すパラメータを用いて，ファイル配布に要する時間 T_{trans} を最小化する，中間ワーカ台数の最適値 w_{opt} を導出する．ここで，P2P 通信の実効通信速度 S_{P2P} は，PeerJS のデフォルトパラメータを用いた予備実験ではおおよそ 0.1[MB/s] であった．

- 全ワーカ台数：W[台]
- 中間ワーカ台数：w ($\leq \lfloor W/2 \rfloor$)[台]
- ファイルサイズ：F[MB]
- マスタ-ワーカ間帯域幅：H_{MW} [MB/s]
- ワーカ-ワーカ間帯域幅：H_{WW} [MB/s]
- P2P 通信の実効通信速度：S_{P2P} [MB/s]

マスタから w 台の中間ワーカへのファイル配布に要する時間 $T_{MW}(w)$[s] は，サイズ F のファイルを w 台に順番に配る時間に等しいため，式 (1) で表される．

$$T_{MW}(w) = \frac{F}{H_{MW}} \times w \tag{1}$$

また，中間ワーカ 1 台から複数の子ワーカへファイル転送が同時に行われると仮定する．中間ワーカと子ワーカ間の帯域幅 H_{WW} が無限大の場合，子ワーカの台数によらず，中間ワーカと各子ワーカとの通信速度はそれぞれ S_{P2P} である．H_{WW} が有限の場合，同時にコピー可能な子ワーカの台数が制限され，その際の平均転送速度は $\frac{H_{WW}}{W-w}$ である．よって，中間ワーカから子ワーカへのコピーに要する時間 $T_{WW}(w)$ は式 (2) で表される．

$$T_{WW}(w) = \begin{cases} \frac{F \times (W-w)}{H_{WW}} & \text{if } (W-w) \times S_{P2P} \geq H_{WW}, \\ \frac{F}{S_{P2P}} & \text{otherwise.} \end{cases} \tag{2}$$

ここで，簡単のため，マスタから中間ワーカ及び中間ワーカから子ワーカへのファイルコピーが平行して行われると仮定する（転送未完了のファイルも送信可能とする）．この時，配布が完了するまでの時間 T_{trans} は $\max(T_{MW}, T_{WW})$ と等しく，式 (3) が成立する．

$$T_{MW}(w_{opt}) = T_{WW}(w_{opt}) \tag{3}$$

よって，中間ワーカ数最適値 w_{opt} は式 (4) のように表される．式 (4) から，最適値 w_{opt} はファイルサイズ F に依存しないことが分かる．

$$w_{opt} = \begin{cases} \frac{H_{MW} \times W}{H_{MW} + H_{WW}} & \text{if } (W-w) \times S_{P2P} \geq H_{WW}, \\ \frac{H_{MW}}{S_{P2P}} & \text{otherwise.} \end{cases} \tag{4}$$

5. 性能評価

5.1 実験概要

4.2 節で実装した，マスタ・ワーカ型ファイル配布機能および P2P 型ファイル配布機能の性能評価として，全ワーカへファイル配布が完了するまでに要する時間を計測する実験を行った．実験は，マスタ・ワーカ・PeerServer を全て 1 台の PC 上で動作させ，マスタ・ワーカ間やワーカ・ワーカ間の帯域制限には shaperd[9] を用いた．評価実験を行った PC の性能などの実験条件を表 1 に示す．

表 1 実験条件

OS	Ubuntu 12.04 LTS 32bit
CPU	Intel Core2 Quad Q6700
メモリ	4GB(実効帯域幅 4.685 [GB/s])
Node.js	ver. 0.10.20
PeerJS	ver. 0.3.8
ブラウザ	Chrome ver. 31

システムに対するワーカの参加方法として，1 秒につき 5 台ずつのワーカが参加する方法を取った．全ワーカ数は，実験が可能な範囲で最大 40 台（タブ数 40）とした．また，配布するファイルのサイズ F は 8MB とした．典型的な VC におけるジョブファイルのサイズは，Asteroids@home などでは 50 - 200kB 程度 [10] であるが，小規模な実験環境を考慮し，また画像処理アプリケーション等を想定して大き目のファイルサイズを設定している．

また，インターネットを用いた大規模な VC システムを想定した場合，多数のワーカが接続するマスタの実効帯域幅は狭くなると考えられる．よって本実験では，マスタ・ワーカ間の帯域幅がワーカ・ワーカ間の帯域幅よりも狭くなるように設定した．全ワーカ台数・中間ワーカ台数を変化させた場合の実験パラメータをそれぞれ表 2 と表 3 に示す．

表 2 全ワーカ台数 W を変化させる場合のパラメータ

ワーカ台数 W[台]	5 - 40
中間ワーカ台数 w[台]	$\lceil \frac{W}{10} \rceil$
ファイルサイズ F[MB]	8
通信帯域幅 [MB/s]	マスタ・ワーカ間 T_{MW} : 2
	ワーカ・ワーカ間 T_{WW} : 6

表 3 中間ワーカ台数 w を変化させる場合のパラメータ

ワーカ台数 W[台]	40
中間ワーカ台数 w[台]	4 - 20
ファイルサイズ F[MB]	8
通信帯域幅 [MB/s]	マスタ-ワーカ間 T_{MW} : 1
	ワーカ-ワーカ間 T_{WW} : 3 or 6

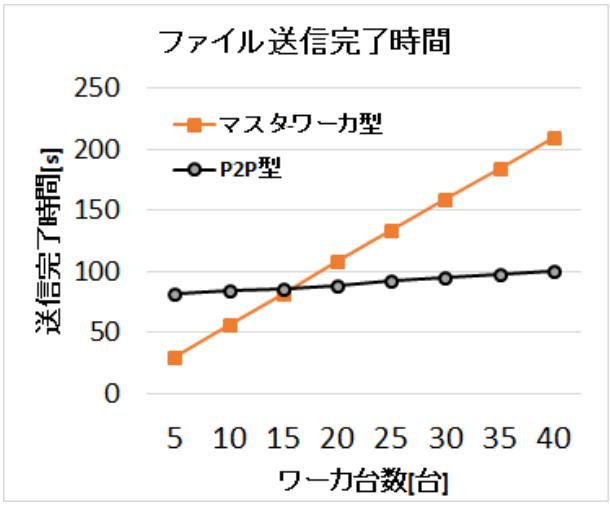


図 7 全ワーカ台数 W を変化させた場合のファイル配布時間

5.2 評価結果

5.2.1 全ワーカ台数 W を変化させた場合

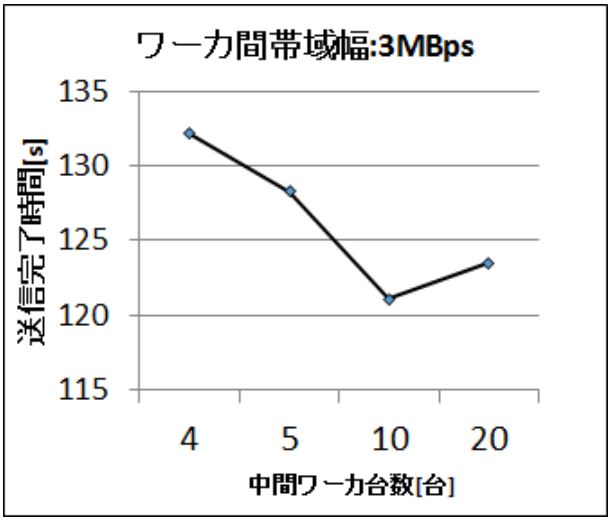
全ワーカ台数を 5 台から 40 台まで変化させた場合の実験結果を図 7 に示す．図 7 から，ワーカ台数が少ない場合は P2P 型の方がファイル配布時間が長いが， ワーカ台数が増えていくにつれて P2P 型よりもマスタ・ワーカ型の配布時間が長くなっていくことが分かる．ワーカ台数が 20 台以上の場合は，P2P 型の方がマスタ・ワーカ型よりも短い時間でファイル配布を完了し，ワーカ台数が 40 台の時は，マスタ・ワーカ型の約半分の時間で配布が完了している．これは，ワーカ台数が少ない時はマスタ・ワーカ型でファイルを配布するのに十分な帯域幅 T_{MW} があるが，ワーカ台数が多くなると，マスタ・ワーカ型では T_{MW} がボトルネックとなり配布時間がワーカ数に比例して増大してしまうのに対して，P2P 型ではこの問題が起きにくく，より多数のワーカの処理が可能となることを意味している．

5.2.2 中間ワーカ台数 w を変化させた場合

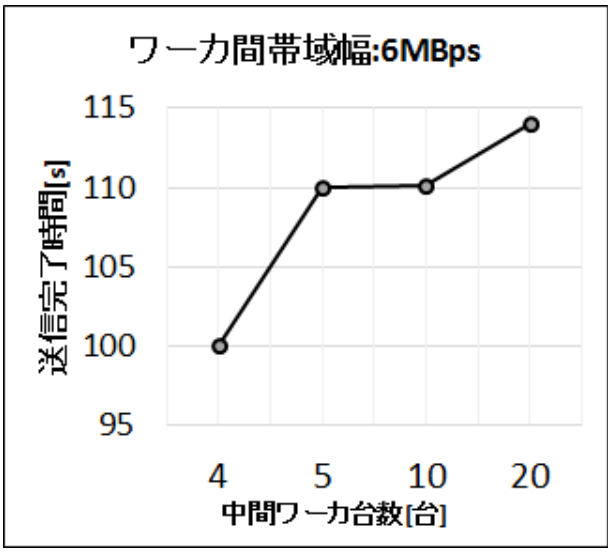
図 8 に，中間ワーカ台数 w を変化させた場合の実験結果を示す．図 8 から，w の値によってファイル配布完了時間が大きく異なり，w に最適値が存在することが分かる．また，w の最適値は，ワーカ・ワーカ間の帯域幅が $H_{WW} = 3$ の時 (図 8(a)) では 10, $H_{WW} = 6$ の時 (図 8(b)) では 4 となり，最適値 w_{opt} が帯域幅に応じて変化することが分かる．

図 8(a) の場合，4.3 節で導出した最適値計算式 (式 (4)) に P2P 通信実効速度 $S_{P2P} = 0.1$ を代入すると $w_{opt} = 10$ となり，実験結果における最適値 10 と一致する．一方，図 8(b) では，式 (4) を用いて導出した値は $\frac{1}{0.1} = 10$ であり，実験結果における最適値 4 とは異なる値となってしまう．

w の最適値について，式 (4) と実際の最適値が異なる理由は，マスタ・中間ワーカ間のファイル転送待ち時間のためと考えられる．式 (4) の導出過程では，転送未完了のファイルも送信可能と仮定しているが，実際にはマスタ・中間ワーカ間の転送が完了し，中間ワーカがファイル送信可能



(a) ワーカ・ワーカ間帯域幅：3MB/s



(b) ワーカ・ワーカ間帯域幅：6MB/s

図 8 中間ワーカ台数 w を変化させた場合のファイル配布時間

となるまで，中間ワーカ・子ワーカ間の転送は待たされる事になる．このファイル転送待ち時間は，中間ワーカ 1 台に対して最大で $\frac{F}{H_{MW}} = 8$ 秒間である．従って，図 8(b) の場合のように，ワーカ・ワーカ間帯域幅 $H_{WW} = 6$ の値が十分大きい場合，少数の中間ワーカで多数の子ワーカに同時にファイル配布ができるため，中間ワーカ数を少なくしファイル転送待ち時間を減らした方が，早く配布が完了すると考えられる．このようなファイル転送待ち時間を含めた w の最適値の導出は，今後の課題の 1 つである．

6. まとめと今後の課題

本研究では，今後の VC システムにおいて必要な機能の一つとしてユーザ間 P2P 通信機能に着目し，これを PeerJS ライブラリを用いて Web ベース VC 上に実装した．この P2P 通信機能の利用例として，巨大なファイルを P2P 通信によってワーカ同士でコピーする，P2P 型ファイル配布機能を実装した．性能評価実験により，ワーカ数が 40 程度の場合，ワーカ全員に直接ファイルを配布するマスタ-ワーカ型と比べて, P2P 型ファイル配布機能を用いることで配布に要する時間を半分程度まで短縮できる事を示した．

今後の課題として，ファイル転送待ち時間を含めた中間ワーカ台数の最適値導出や，P2P 通信機能を利用した Web ベース VC 上の負荷分散機能の実装が考えられる．また，今回の実験ではワーカ数が 40 程度と少ないため，複数の計算機を用いて規模の大きい性能評価実験を行った後，実際に Web ベース VC として運用しシステム全体の性能評価を行う予定である．

参考文献

[1] SETI@home: http://setiathome.berkeley.edu.
[2] BOINC: http://boinc.berkeley.edu.
[3] asm.js: http://asmjs.org.
[4] PNaCl : https://developers.google.com/native-client/dev/.
[5] Johnston, Alan B., and Daniel C. Burnett, "WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web", Digital Codex LLC, 2012.
[6] Pimentel, Victoria, and Bradford G. Nickerson, "Communicating and displaying real-time data with WebSocket", Internet Computing, IEEE, Vol.16, Issue 4, pp.45-53, 2012.
[7] PeerJS: http://peerjs.com/
[8] Node.js: http://nodejs.org/
[9] shaperd : http://bto.la-terre.co.jp/support/shaperd.html
[10] Asteroids@home: http://asteroidsathome.net/boinc/
[11] D. Kondo, F. Araujo, P. Malecot, P. Domingues, L. M. Silva, G. Fedak, and F. Cappello, "Characterizing Error Rates in Internet Desktop Grids", 13th European Conf. Parallel and Distributed Comput., pp. 361–371, 2007.
[12] D. Kondo, Javadi, B., Malecot, P., Cappello, F. and Anderson, D.P., "Cost-benefit analysis of Cloud Computing versus desktop grids", *IPDPS*, pp. 1–12 (online), 2009.
[13] 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, vol.49, no.3, p1214-1228, 2008.
[14] 高木 省吾, 渡邊 寛, 福士 将, 天野 憲樹, 舩曳 信生, 中西 透, "Web ブラウザを用いたボランティアコンピューティングプラットフォームの提案", 情報処理学会研究報告 2014-HPC-143(29), pp. 1–8, 2014.
[15] C. Lattner and V. Adve, "LLVM: A Compilation Framework for Lifelong Program Analysis & Transformation", Proc. of the 2004 International Symposium on Code Generation and Optimization (CGO'04), pp.75–86, 2004.
[16] K. Watanabe, M. Fukushi and S. Horiguchi, "Expected-credibility-based Job Scheduling for Reliable Volunteer Computing", IEICE Trans. Inf.& Syst., Vol.E93-D, No.2, pp.306 – 314, 2010.