RF-006

既存 Web ページ同期編集機構 WFE-S のクラウド環境への適用について
A Synchronous Web Page Editing System WFE-S based on Cloud Computing Environment

井上 良太† 加藤 雄大† 合田 拓史†
Ryota Inoue Yudai Kato Takushi Goda
白松 俊‡ 大囿 忠親‡ 新谷 虎松‡
Shun Shiramatsu Tadachika Ozono Toramatsu Shintani

1. はじめに

Web 上での協調作業に関する研究は多数行われており，Web インテリジェンス技術におけるこの課題に関する研究は重要である[1]．本研究室においても既存 Web ページ同期編集機構 WFE の開発を行なってきた [2][3]．WFE では，システムを導入した Web サーバ上に存在する Web ページを，ブラウザ上から編集することを可能にしている．さらに，複数クライアントで，変更内容をリアルタイムに表示しながら，Web ページを同時に編集するといった協調作業を可能にしている．このような協調作業を本研究では同期編集と呼ぶ．図 1 に WFE による同期編集の様子を示す．WFE の編集機能には，文字の変更，コメントの付与，画像の追加，テーブルの編集などがある．これらの機能は，編集箇所で右クリックをすることで表示されるメニューから実行できる．

本稿では， この WFE をクラウドコンピューティング環境（以降，クラウド環境と略す）に適用する実装手法，および，その利点について述べる．本研究では，クラウド環境として，Google App Engine（以降，GAE と略す）を利用した．GAE とは，Google が提供する Web アプリケーション開発支援機構である．Google のインフラストラクチャを使用した Web アプリケーションを開発できるため，ハードウェア管理の必要性が無いクラウドアプリケーションを容易に作成可能であることが最大の利点となっている．以降，本稿では，GAE アプリケーションとして既存 Web ページの同期編集機構を再構築したシステムを WFE-S と呼ぶ．WFE-S では，同期編集における HTML コンテンツの管理をクラウド環境 で行い，動的なコンテンツを含まないあらゆる Web ページを対象とし，同期編集可能とすることを試みた．本システムは，既存の Web ページを複数人でリアルタイムに編集して共有するという，Web の新たな利活用方法を確立する基盤技術として発展していくことが期待できる．

また，本研究では， 先行研究のシステム WFE における課題の解決も試みた．2 節で詳しく述べるが WFE には同期におけるリアルタイム性と，システムの導入に関して課題があった．WFE-S では，これらの課題を解決している．

本稿の構成について述べる．2 節では先行研究のシステムの問題点を示す．3 節では，本システムの構成を述べ，2 節で述べた問題が本システムにより解決可能であることを示す．4 節では，通信に用いた機構について詳細に述べる．5 節では，本システムの利用方法と導入部分の実装について述べる． 6 節で評価と考察を示し，7 節でまとめとする．

†名古屋工業大学工学部情報工学科 Department of Computer Science, Engineering, Nagoya Institute of Technology.
‡名古屋工業大学大学院情報工学専攻 Dept. of Computer Science and Engineering, Graduate School of Engineering, Nagoya Institute of Technology.

図 1:WFE の実行例

2. WFE における課題

本節では，先行研究のシステム WFE を用いた Web ページの同期編集における課題について述べる．

2.1 リアルタイム性

WFE では，クライアントサイドからのポーリングによりサーバ上の HTML ファイルが変更されたことを検出して同期を行なっていた．このため，あるクライアントによって送信された変更内容が，全クライアントに反映されるまでの時間に，クライアントごとの誤差が発生していた．この誤差を本稿では同期誤差と呼ぶ．本システムを含む Web ページ同期編集機構では，複数クライアントによる編集がほぼ同時に確定すると，編集が衝突して編集内容が喪失してしまうという課題がある．この編集内容の喪失は，クライアント間の同期誤差が大きくなるにつれて，発生する可能性が高くなる．これはポーリングの間隔を短くすることで解決できる可能性があるが，サーバへの負荷が大きくなり，スケーラビリティの低下につながる．

2.2 導入の困難さ

WFE では，編集対象の Web ページを配信するサーバを構築し，Web サーバに対してアプリケーションをインストールする必要があった．このようなシステムの利用形態では，同期編集可能な Web ページが自身で管理しているものに限定されてしまう．また，サーバ管理に不慣れな人にとって不向きである．例えば，サーバ管理を他社に任せている企業において，自社の Web ページをデザインし直す際に，現在の Web ページを変えながら検討していきたい場合，アプリケーション導入のためにはサーバの管理会社に問い合わせるという煩雑な作業が必要となってしまう．

3. クラウド環境への実装手法

本節では，Web ページの同期編集機構を，GAE を用いたクラウド環境として構築する手法について述べる．さらに，2 節で提示した先行研究の問題点を解決する手法について述べる．

3.1 システム構成

図 2 は，WFE-S のシステム構成図である．本システムは，編集ページ確認・生成モジュール，編集ページ HTML 取得モジュール，および，更新・プッシュ配信モジュールから構成される．ここでの編集ページとは，ブックマークレット呼び出し時閲覧中の Web ページとは別に生成される，編集機能を持った GAE 上の Web ページである．編集ページ上ではリアルタイムに他クライアントによる編集結果を確認でき，リアルタイムに自分の編集結果が他クライアントに反映される．

本システムは，ブックマークレットを Web ページの編集に使用するブラウザのブックマークに登録することで利用できる．ユーザがブックマークレットを呼び出すと，まず GAE のサーバ上の編集ページ確認・生成モジュールへ閲覧中の Web ページの URL が送信される．編集ページ確認・生成モジュールでは，受信した URL から対応する編集ページの URL をデータストアから取り出すか，あるいは，新規生成して返す．このとき返される編集ページの URL は”http://wfesyncroshare.appspot.com/(編集ページのファイル名)”という形式になっている．編集ページのファイル名は新規生成時に編集ページの HTML に対して与えられるユニークな ID である．新規生成時には，編集ページ確認・生成モジュールは，受信した URL から Web ページの HTML を取得して同期編集可能な HTML に変更し，ユニークな ID をファイル名として与えてデータストアに保存する．また，本システムは，動的に Web ページを書き換える JavaScript の実行を対象外とするため，GAE サーバ上に Web ページを取り込む段階で HTML 内の script タグを削除する．同期編集可能な HTML への変更の詳細は 5 節で述べる．

ブックマークレットが，編集ページの URL を受け取ると，編集ページの URL を GAE のサーバ上の編集ページ HTML 取得モジュールへ送信する．編集ページ HTML 取得モジュールは受信した編集ページ URL を使ってデータストアを参照し，編集ページ HTML を取得して返す．この結果，ブックマークレットの呼び出し操作のみで自動的に編集ページへ遷移して，Web ページの同期編集を開始することが可能となる．

編集ページ上では，先行研究のシステムで利用可能であった文字列の変更，テーブルの編集，リストの編集，コメントの追加等の機能により Web ページを編集することができる．ユーザが Web ページの編集を行うと，更新された HTML が GAE のサーバ上の更新・プッシュ配信モジュールへ送信される．更新・プッシュ配信モジュールは受信した更新 HTML をデータストアに保存して，更新に必要な body タグ内の一部部分を全クライアントへプッシュ配信する．クライアントが受け取った変更情報で HTML を書き換えることで，表示されるページの同期が実現される．

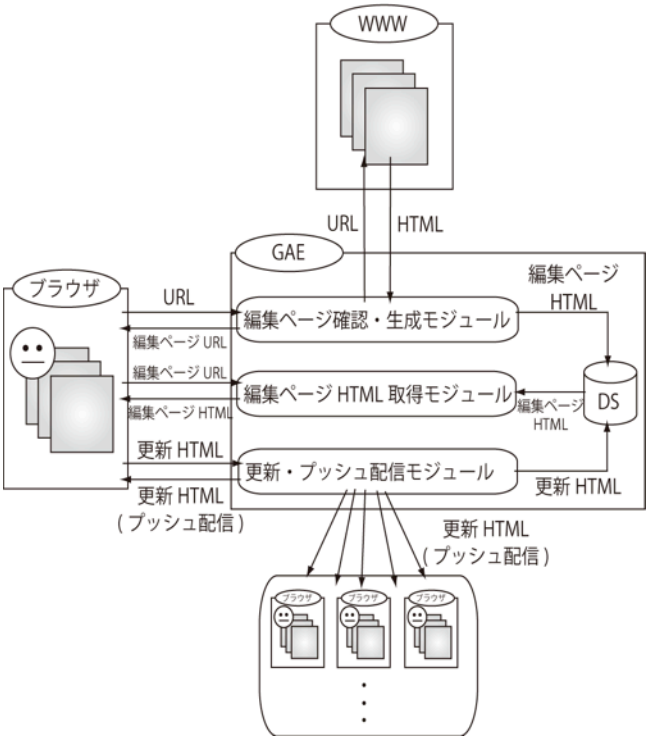

図 2:システム構成図

3.2 リアルタイム性の向上

2 節で述べた，リアルタイム性に関する課題の解決方法を示す．WFE-S では，GAE が提供するプッシュ配信のための機構である Channel API を使用して，プッシュ配信によるページの同期を行う．Channel API では，クライアントはそれぞれチャネルと呼ばれる通信路をサーバとの間に構築し，通信を行う． 本システムでは，プッシュ配信先を特定するために，サーバはチャネルを編集ページごとにグループ化して管理する必要がある．詳細な実装方法は 4 節で述べる．

ポーリングによる同期では各クライアントがそれぞれのタイミングでサーバへ更新を確認するに対し，プッシュ配信による同期ではサーバから全クライアントへ一斉に変更内容が送信されるので，同期編集における同期誤差は軽減される．

3.3 ブックマークレットによる簡易な導入

2 節で述べた，導入の困難さに関する解決方法を述べる．WFE-S では，ブックマークレットにより簡易に同期編集を開始することができる．ブックマークレットは，閲覧中の Web ページを GAE サーバへ取り込み，編集機能を持たせた Web ページを GAE サーバ上に新規生成する作業と，生成した Web ページへ遷移する作業を自動化している．ユーザには，元の Web ページの GAE サーバへのコピーや，転送処理などが効果的に隠蔽されることで，容易な導入を可能にしている．

また，同期編集は，ブックマークレットで生成した GAE サーバ上の Web ページで全て行われるため，サーバ管理の煩雑さが解消されている．

4. プッシュ配信機構

プッシュ配信には，GAE のプッシュ配信のための Channel API を使用した．本節では，本システムにおけるプッシュ配信機構の実装方法を詳細に述べる．また，Channel API を使用する際の本システムにおける課題と解決策を述べる．

4.1 プッシュ配信機構の実装

図 3 にプッシュ配信機構のシーケンス図を示す．プッシュ配信機構は，チャネルの構築と変更内容の通知という二つの段階に大きく分けられる．チャネルの構築は，サーバ-クライアント間の通信路の確立段階である．変更内容の通知は，編集が行われた場合の実際のプッシュ配信を行う段階である．図 3 におけるユーザ A とユーザ B は，同一編集ページ上で同期編集を行うユーザとする．client.js は，それぞれのクライアントにおいて，プッシュ配信機構のクライアントサイドの処理を行う JavaScript ファイルを示す．また，ユーザ B はチャネルの構築が終了しているものとする．

クライアントは，編集ページへの遷移が完了すると，サーバとの通信路であるチャネルを構築する．チャネルの構築には，識別子としてサーバで発行されるトークンが必要である．トークンは，GAE サーバ上のトークン発行機構へ Ajax 通信により発行要求を出すことで取得する．トークンは，プッシュ配信先を特定するために，編集ページごとにグループ化する必要がある．このため，図 3 におけるユーザ A の client.js のように，発行要求と同時に編集ページのファイル名も送信する．トークン発行機構は編集ページのファイル名に関連付けられたトークンを発行することで，トークンのグループ化を実現する．具体的には，受け取った編集ページのファイル名を Channel API の create_channel メソッドの引数に指定する．クライアントがトークンを受け取ると，トークンを使ってチャネルを生成し，ソケットを開いて変更内容を待ち受ける．

クライアント側のソケットにはコールバック関数を定義する．本システムではソケットのコールバック関数の内，プッシュ配信によるメッセージ到着時に呼ばれる onmessage を使用している． onmessage では，変更内容を受け取った際に，その変更内容を用いたページ書き換え処理を行う．

サーバからのプッシュ配信は send_message メソッドにより行う．send_message メソッドは，第一引数に指定された ID と関連付けられたトークンを持つチャネルにのみ第二引数のメッセージを送信する．トークン生成時に編集ページのファイル名と トークンを関連付けているため，send_message の第一引数には編集ページのファイル名を指定し，第二引数には変更内容を指定することで，その編集ページのチャネルを持つクライアントに対してのみ変更内容を送信することを可能にしている．

4.2 通信用 iframe 要素

Channel API を使用する場合に，クライアントサイドでは，通信に必要なクラスやメソッド，動作を定義している Channel API のための JavaScript ライブラリを読み込む必要がある．このライブラリは，通信に用いる API を HTML

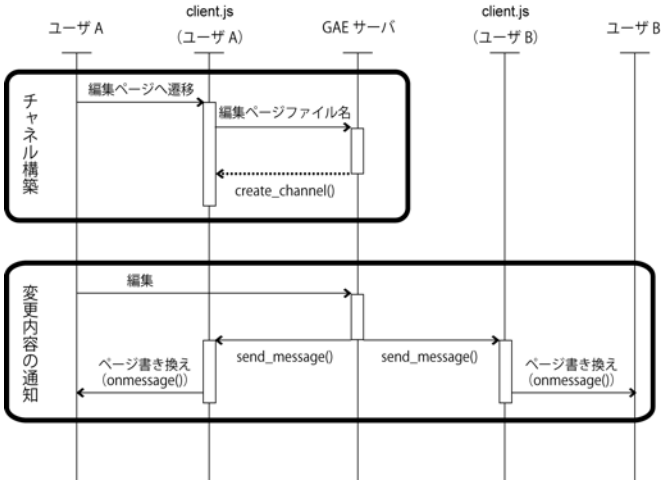

図 3:プッシュ配信機構

```
<iframe name="wcs-iframe" id="wcs-iframe" style="height: 100%; width: 100%; "
src="http://950.talkgadget.google.com/talkgadget/d?token=AHRlWrp2GUxHTO3
1mJhMGd9jCtrpkTVEsGVLRBPjLL9rsw6sKZZoW4_zf3J6ilPzfuvvQUBTpi1K40AC1zG
GN9lGmRXeyHlQSvLSo6Dl9eJBtPB8zE7pjjODEQ-uHDOxU1eZk0zkfNBuKrbcZAoW
3L8f204LVVln-g&amp;xpc=%7B%22cn%22%3A%22DYj2YBz0bB%22%2C%22tp%
22%3Anull%2C%22ppu%22%3A%22http%3A%2F%2Fwfesynchrosharer.appspot.
com%2F_ah%2Fchannel%2Fxpc_blank%22%2C%22lpu%22%3A%22http%3A%2
F%2F950.talkgadget.google.com%2Ftalkgadget%2Fxpc_blank%22%7D"></ifram
e>
```

図 4: 通信用 iframe 要素の例

に組み込む iframe 要素(図 4)を， body 要素の末尾に生成する．以降，この iframe 要素を通信用 iframe 要素と呼ぶ．

通信用 iframe 要素では，具体的に，API “http://talkgadget.google.com/talkgadget/d” を，パラメータにチャネルのトークンなど，通信に必要な情報を指定して呼び出す．API を呼び出すと，結果として新たな JavaScript ファイルが得られ，この JavaScript ファイルによって通信が確立される．通信に用いる静的な情報を保持している通信用 iframe 要素の扱いは，HTML を書き換えるという本システムの特徴から重要な課題となる．以降，本節では，通信用 iframe 要素に関する課題と解決策を述べる．

当初，本システムでは，クライアントがページ内容を変更して変更後の HTML がサーバへ送信されると，サーバサイドで受け取った HTML をそのまま全クライアントへ配信していた．配信された HTML でページの書き換えを行うと，全クライアントの通信用 iframe 要素は変更を行ったクライアントの通信用 iframe 要素に統一される．この結果，通信がうまくいかなくなるという問題があった．

この問題点の解決策を説明する．まず，GAE サーバ上の編集ページ確認・生成モジュール(図 1)が，編集対象の Web ページの HTML ファイルをデータストアへアップロードする前に， body の子要素を全て含むような div タグを挿入する．この結果，編集ページでは，body 内は大きく div 要素と通信用 iframe 要素の二つに分割されることになる．そして，クライアントでの HTML の書き換え部分を div タグに限定することで，通信用 iframe 要素が書き換えられることを回避している．

5. システムの利用とブックマークレットを用いたシステムの導入

本節では，まず，ユーザの視点からシステムの利用の流れについて述べる．その後，システムの導入に用いるブックマークレットの詳細な実装について述べる．

5.1 システムの利用の流れ

本システムのユーザは，編集対象の Web ページ上でブックマークレットを呼び出すことで，編集対象の Web ページに対応する編集ページへと遷移する．編集ページへ遷移後，ユーザが同期編集を開始するにはログインを行う必要がある．編集ページに入り右クリックをするとログインのためのボタンが表示され，ボタンを押すことで図 5 に示すパスワード入力プロンプトが表示される．ここで入力するパスワードは編集ページ作成要求時に作成者が設定したパスワードである．入力されたパスワードは，編集ページのファイル名と共にサーバの認証機構へ送信される．サーバでは受け取った編集ページのファイル名を用いてデータストアを参照し，編集ページのパスワードを取り出して認証要求時に受け取ったパスワードと比較する．一致する場合には認証成功を，一致しない場合，あるいは，認証要求時に受け取ったファイル名の編集ページがデータストア上に存在しない場合には認証失敗を，それぞれ HTTP ステータスコードを用いて通知する．クライアントは認証成功の通知を受け取ると，編集機能に必要な JavaScript 群を順に読み込むことで編集可能な状態とする．一方，認証失敗の通知を受け取ると，再認証のためにページのリロードを要求する．

編集可能な状態となると文字の変更，コメントの挿入，タイトルの変更，テーブルの編集，リストの編集といった先行研究における編集機能により Web ページを編集することが可能になる．

同期の様子を図 6 に示す．あるユーザが編集を行って確定すると，編集者の編集ページの HTML が変更に合わせて書き換えられ，書き換えられた HTML 全体が変更内容としてサーバへ送信される．サーバは受け取った HTML から通信用 iframe 要素を削除してデータストアへ保存し，title タグと body タグ内の書き換え部分の div タグの内容を取り出して全クライアントへプッシュ配信する．

各クライアントは，受け取った title タグと div タグの内容を用いて編集ページの HTML を書き換えることでページの同期を行う．この結果，各ユーザはリアルタイムに編集者によって行われた変更を確認することができる．

5.2 ブックマークレットに基づくシステム導入手法

ブックマークレット呼び出し時の動作の詳細を図 7 に示す．ブックマークレット呼び出し時の処理は，データストア確認段階と編集ページ生成段階に分けられる．

データストア確認段階では，閲覧中の Web ページの URL を GAE サーバの編集ページ確認機構へ送信し，データストア上に閲覧中の Web ページに対応した編集ページが存在するかどうかを確認する．ブックマークレットを呼び出すと，クライアントから閲覧中のページの URL がサーバに送られる．サーバは，受信した URL の編集ページがデータストア上に存在するかどうかを確認して，結果をクライアントサイドへ返す．このとき返される結果は，編集ページが存在

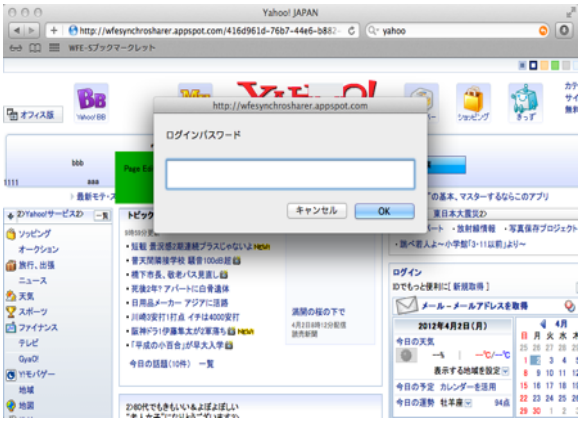

図 5:パスワード入力画面

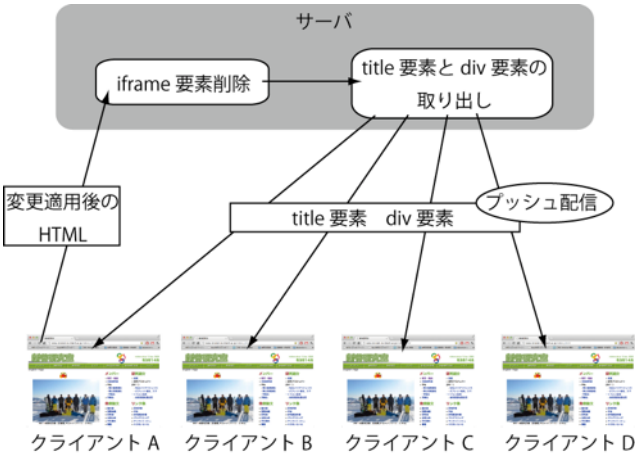

図 6:同期の様子

する場合は編集ページのファイル名となり，存在しない場合は存在しないことを通知する文字列となる．

クライアントでは，編集ページがデータストア上に存在した場合，編集ページのファイル名を受信して編集ページへ遷移する．編集ページがデータストア上に存在しなかった場合は，次の編集ページ登録段階に移る．

クライアントで，編集ページがデータストア上に存在しないことを通知する文字列を受け取ると，編集ページに設定するパスワードを入力するプロンプトが表示される．このパスワードは，編集可能なユーザを制限するための認証に用いられる．パスワードが正しく入力されると，閲覧中の Web ページ（以下元の Web ページ）の URL とホスト情報，元の Web ページのエンコード形式，生成要求を出したクライアントのブラウザのユーザエージェント，パスワードを GAE サーバの編集ページ生成機構へ送信する．

生成要求を受け取った編集ページ生成機構は，はじめにユニークな ID を生成する．生成した ID は編集ページのファイル名として用いられる．次に，受け取った元の Web ページの URL に対して，HTTP リクエストを発行して HTML を取得する．このとき元の Web ページが非対応のブラウザに対して簡易ページを用意している可能性があるので，HTTP ヘッダの User-agent に受け取ったユーザエージェントを指定する．

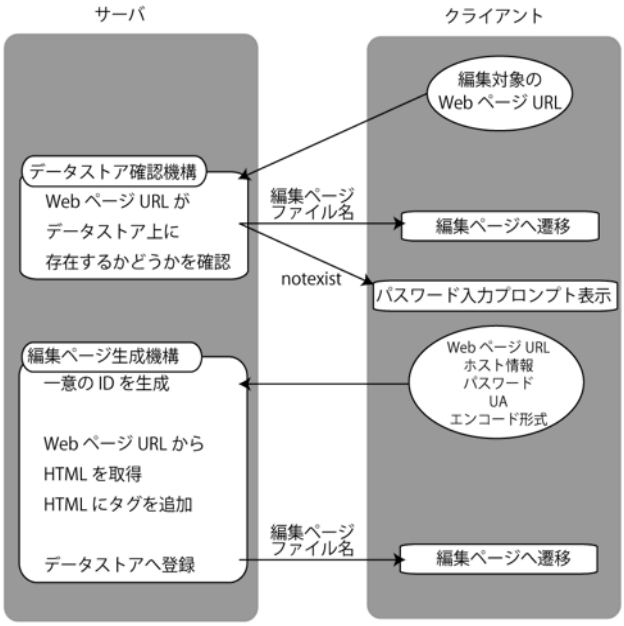

図 7:ブックマークレットの動作詳細

元の Web ページの HTML を編集可能な Web ページとするために，HTML へ 3 種類のタグを付加する．一つ目は，base タグの付加である．編集ページは GAE のサーバ上に設置されるため，元の Web ページの HTML 内に存在する相対パスが機能しなくなる．この問題を解決するために base タグを元の Web ページの head 内に付加する．付加される base タグの href 属性には，編集ページ生成要求時に受け取った元の Web ページのホスト情報を指定する．二つ目は，body タグ内の書き換え部分と，通信用 iframe 要素を分離する div タグの付加である．通信用 iframe 要素は通信に必要な情報を保持しているため，書き換えることができない．よって，書き換え部分と通信用 iframe 要素を分離する必要があり，div タグを付加することでそれを実現する．三つ目は，同期編集機能を持たせるためのスクリプトタグの付加である．ここで付加されるスクリプトタグは，クライアント側で Channel API を利用するために必要な Channel API の JavaScript ライブラリ，各編集機能を持たせるための JavaScript，Channel API によって通信を行う際の動作を記述した JavaScript である．

生成した ID を key_name として元の Web ページの URL，編集ページの HTML，編集ページ作成要求時に受け取ったパスワードをデータストアへ新規登録する．編集ページの HTML の登録にはエンコード形式を指定する必要があるので，編集ページ作成要求時に受け取ったエンコード形式をここに指定する．

最後に，クライアントへ編集ページのファイル名を返す．クライアントは，受信したファイル名を用いて編集ページへ遷移する．

6. 評価と考察

本システムは，遠隔地にいる数人のユーザがブラウザ上で Web ページの同期編集を行うということを想定している．遠隔地に存在するクライアントが同期編集を行う際には，編集の衝突を避けるために全クライアントについて同期誤差を最小とすることが要求される．そこで本実験では，レスポンスを測定することで本システムでの同期精度を評価する．本実験におけるレスポンスとは，あるユーザが編集を行って変更内容がサーバへ送信された時点から，他のクライアントで変更内容を受け取って変更の適用を完了する時点までの時間と定義し，同期精度とはクライアント間でのレスポンスの誤差の程度と定義する．レスポンスの測定は先行研究のシステム WFE と本システム WFE-S において行い，比較することで本システムの同期精度を明示する．

6.1 実験方法

実験方法の確立においては，Ajax 関連のテスト手法を参考にした[4]．想定ユーザ数は 5 人とし，1 つのクライアントが 5 秒間隔で，Web ページ上の文字をランダムに 100 回更新して，他 4 つのクライアントでレスポンスを算出して記録する．ここでのクライアントは同一計算機上で起動されているブラウザとする． この計算機の仕様は，CPU が 2.7GHz Intel Core i5，メモリ 4GB のデスクトップ PC である．また，先行研究のシステムにおける実験では，サーバとしてローカルネットワーク上に存在する同一のデスクトップ PC を使用した．さらに，サーバ・クライアント間の通信遅延を 1000 回の ping により測定した結果，ローカルネットワーク上では平均が 8.698ms，分散が 10.330 であり，GAE では平均が 45.091ms，分散が 4.348 であった．

レスポンスを測定するためには測定開始時刻と終了時刻の取得を行う必要がある．本実験では時刻の取得は全てクライアント上で行うが，クライアントは同一計算機上で起動されているブラウザであるので時刻は同期されているものとする．測定開始時刻として，ランダム編集が行われるクライアント（以下，編集者クライアント）が変更内容を送信する直前に時刻の取得を行う．また測定終了時刻として，変更内容を受け取ったクライアント（以下，受信者クライアント）が HTML の書き換えを完了したところで時刻の取得を行う．先行研究のシステムにおける実験では，測定開始時刻は編集者クライアントが送信する変更内容の HTML にメタタグとして付加される．受信者クライアントはポーリングにより取得した変更内容の HTML から，メタタグを取り出すことで測定開始時刻を得て，測定終了時刻の差をとってブラウザのコンソールに出力する．本システムにおける実験では，測定開始時刻は編集者クライアントから変更内容と同時にサーバへ送信されて プッシュ配信される．受信者クライアントは，プッシュ配信された測定開始時刻と HTML 書き換え後に取得した測定終了時刻の差をとってブラウザのコンソールに出力する．

4 つの受信者クライアントでコンソールに記録されたレスポンスについて，編集内容ごとに標準偏差を算出する．標準偏差は受信者クライアントにおける同期のタイミングの誤差を表しているため，標準偏差を観察することで同期精度を確認することができる．

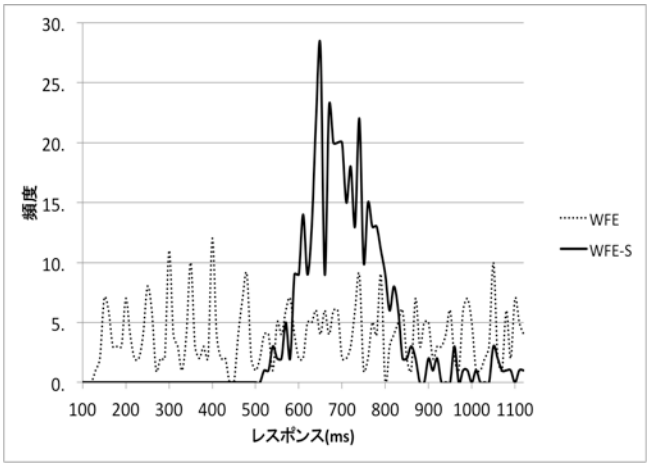

図 8:レスポンスの分布

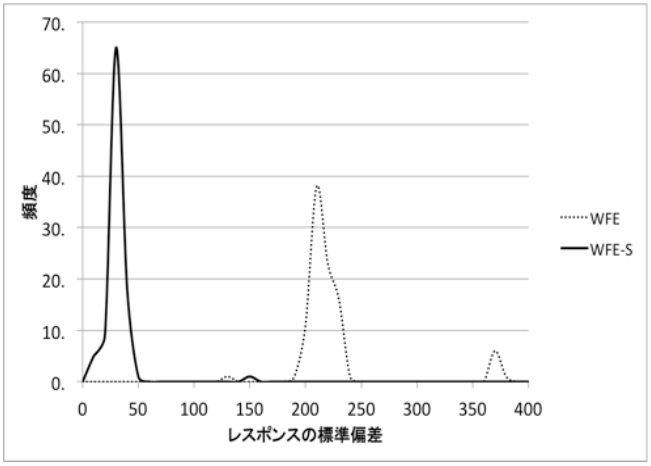

図 9:レスポンスの標準偏差の分布

6.2 実験結果

100 回の測定実験で得られた，レスポンスの分布を図 8 に示す．横軸はレスポンスを表しており，縦軸は頻度を表す． 図 8 により，WFE のレスポンスは広範囲にわたって分布しているが，WFE-S のレスポンスの多くは 500～900ms 付近に分布していることが分かる．実際には WFE-S のレスポンスの 96.25%が 520～920ms に分布している．

図 9 はレスポンスの標準偏差を示している．横軸は更新ごとのレスポンスの標準偏差を表しており，縦軸は頻度を表す． WFE-S では標準偏差が 10～50 付近に集中しており，レスポンス時間のばらつきが少ないため，ユーザにとって使いやすい．一方，WFE では標準偏差が 190～240 に分布しており，レスポンスのばらつきが大きいため，編集結果が反映されるタイミングが予測しづらくユーザに取って扱いにくいといえる．

6.3 考察

先行研究のシステムではポーリングによりクライアントの同期を行っていた．複数人での同期編集において重要となるのは同期精度であるが，同期精度を改善するためにポーリング間隔を短くすると，サーバへの負荷が増大してスケーラビリティが低下するという問題があった．

本システムは，GAE を利用してアプリケーションを開発することで Channel API を用いたプッシュ配信による同期で同期編集機構を実現して先行研究の問題を解決した．図 9 の結果は Web ページの同期編集におけるクライアントの同期をプッシュ配信機構によって行うことの有効性を表している．レスポンスの標準偏差とは同期精度であり，同期精度が改善されることで編集の衝突が発生する確度が低くなり，編集の衝突による編集内容の喪失が起こる可能性を低減させることができた．

7. おわりに

本研究では，既存 Web ページの同期編集機構を GAE アプリケーションとして再構築し，ユーザを限定しない Web アプリケーションとしての同期編集機構を実現した．また，プッシュ配信を用いた同期方法の確立による，クライアントサイドの同期精度の改善とその評価実験を行った．さらに，スケーラビリティに関しても，GAE のインフラストラクチャを用いたことで，大きく改善されることが期待できる．この点についての評価実験も今後行う予定である．

本システムでは，同期精度を高めて編集が衝突する可能性を低減することができたが，まだ複数のクライアントが同時に編集 を確定した場合には編集の衝突が生じて変更内容が喪失するという問題がある．編集の衝突の可能性をさらに改善するアプローチとして以下の 2 つの部分更新技術を考えており，既に実装段階に入っている．1 つ目は，編集ページの HTML をブロック要素に分割して変更部分のみをやり取りすることで，変更部分のみの更新を実現するアプローチである．2 つ目は，変更部分の DOM 要素の内容とその DOM 要素の XPath を交換することで，XPath を用いて変更部分のみの更新を実現するアプローチである[5]．これらのアプローチの内どちらが有効であるかを評価実験により比較検討していく予定である．

上記の分割・差分更新にはもう一つの目的がある．サーバからのプッシュ配信に用いる Channel API の send_message メソッドには送信容量に 32KB までという制限がある．このため，編集ページの HTML を全て送信する現在のシステムでは，同期編集可能な Web ページが限定されている．本システムの実用性を高めるためにも分割・差分更新の実現は重要な課題である．

参考文献

[1] Max Goldman, Greg Little, Robert C. Miller, “Real-time collaborative coding in a web IDE”, UIST ’11 Proceedings of the 24th annual ACM symposium on User interface software and technology, (2011)

[2] 田代慎治，大囿忠親，伊藤孝行，新谷虎松，“WFE における協調的な Web ページの編集について”，第 67 回情報処理学会全国大会論文集，Vol.2，pp.419-420 (2005).

[3] 土井 達也，白松 俊，大囿 忠親，新谷 虎松，“ブックマークレットを用いた既存 Web ページのリアルタイム編集機構”，情報処理学会全国大会， (2012).

[4] Engin Bozdag, Ali Meshah, Arie Van Deursen, “Performance testing of data delivery techniques for AJAX application”, Journal Web Enginnering, Vol.8, Issue 4, (2009)

[5] 合田 拓史，井上 良太，加藤 雄大，白松 俊，大囿 忠親，新谷 虎松，“Web ページ同期編集システム WFE における差分同期機構の試作”， FIT2012, (2012) (掲載予定).