Vol.2012-DPS-153 No.5
2012/11/15

Web ベースプログラミング環境での インタラクティブ入出力方法の実装

近藤佳代† 柳澤秀明†

近年，クラウドコンピューティングと呼ばれるシステム形態に注目が集まっている．クラウドコンピューティングを使用するメリットは，データや作業環境を複数人で共有することが可能となることである．チームでシステム開発を行う場合，バージョンを統一させた同じソフトを用意する必要があり，パソコンを変更した際には，ソフトの導入から環境の設定まで全て始めから行わなければならない．このため，時間やコストの無駄が発生する．そこで，本研究室では，プログラミング環境を構築する際の問題を取り除くため，Web 上にプログラミング環境を実装している．現状では，バッチモードにのみ対応しており，ユーザからの入力を求めるプログラムの実行ができない．本研究では，これらの問題を解決するためにインタラクティブモードを実装し，ユーザからの入力を求めるプログラムの実行を可能にした．

Implementation of Interactive I/O for Web-based Programming Environment

KAYO KONDOU† HIDEAKI YANAGISAWA†

System configuration called cloud computing has attracted our attention. The merit of using this is that some people can share the data and work environment. If they develop a system in a team, you must prepare the same software of the unified version. In addition, if they change a PC, they must install of the software, set of the environment and so on again. That will be a waste of time and money. So in this research, we have implemented the programming environment on the web to solve problems which come up when we structure the programming environment. This system couldn't run a program that receives input from the user, because it only supported in batch mode. To solve this problem, we have implemented interactive mode. In consequence, we can run a program that receives input from the user.

1. はじめに

近年，クラウドコンピューティングと呼ばれるシステム形態に注目が集まっており，インターネットなどのネットワークを通じてサーバが提供するサービスを利用することで，いつでもどこでも同一環境を利用することができる．

クラウドという形態は様々な場面で利用される．例えば，大規模なシステム開発を行う場合に必要となるのが協調開発環境である．協調開発環境では，バージョンを統一させた同じソフトを用意する必要がある．バージョン管理が正しく行われない場合，バグやセキュリティホールの要因となる可能性があるからである．また，開発途中に使用していたパソコンを変更すると，ソフトの導入から環境の設定まで全て始めから行わなければならない．このため，時間やコストの無駄が発生する．開発の際にクラウド環境を利用すると，OS やアプリケーションのアップグレードもサーバ側で実装すれば良いだけなので，チーム全員が常に最新で同じバージョンを使用可能となり，環境構築の際の負担もかからない．このように，クラウド環境によってリソースを共有することで開発効率を上げることが可能となる．

そこで，本研究室では，協調開発を行う際のプログラミング環境構築時の問題を取り除くため，Web ブラウザ上でのプログラミング環境の開発をしている．これにより，インターネットに接続できる環境とブラウザさえあれば，プログラミングが可能となる．

クラウドコンピューティングの実現方法として，WebOS や VNC(Virtual Network Computing)などがある．WebOS は，デスクトップライクな環境をブラウザ上で提供し，クライアント環境に依存しないため，ユーザは Web ブラウザと WebOS へのネットワーク接続環境だけで各種アプリケーションが利用可能となる．一方，VNC はネットワークを通じて接続された他のコンピュータの画面を遠隔操作するソフトウェアである．VNC では，サーバ OS 上で動作するアプリケーションを利用することが可能であり，ユーザはサーバ OS が提供するデスクトップ環境を利用できる．しかしその反面，VNC ではサーバの負荷が比較的高くなるという短所がある．WebOS では，サーバ OS 上で動作するネイティブなアプリケーションやデスクトップ環境は利用できないが，処理の一部をクライアント側で実行するため，VNC と比較するとサーバの負荷は小さくなる．協調開発を行う際には利用者が多いため，サーバの負荷が重要視される．以上のことより，本研究では Web ベースシステムの実装を行っており，CUI と GUI プログラムの両方に対応しているが[1]，CUI についてはバッチ処理プログラムのみしか実行できなかった[2]．そこで，より柔軟な入力処理を行う

† 徳山工業高等専門学校 専攻科 情報電子工学専攻
Computer Science and Electronics Engineering , Advanced Course , Tokuyama College of Technology

ため，ユーザからの入力によってリアルタイム処理を行うインタラクティブ入出力に対応させる方法について検討した．

本論文では，ブラウザ上のコマンドプロンプトでのインタラクティブ入出力を実現するため，JavaServlet によるマルチスレッド化と非同期通信による実現方法について述べる．

2. 関連研究

Web ベース開発環境として，コーディングのためにはエディタ，ファイルやフォルダ確認のためにはファイルエクスプローラ，コンパイルや実行のためにはコマンドプロンプトなどが必要となる．これらの関連研究として，ターミナルでは WebTTY[3]や Anyterm[4]，テキストエディタでは TextDrop[5]や ACE[6]，統合開発環境では Cloud9 IDE[7]や Monaca[8]，VNC を利用したシステムでは VM ware View[9]や DEVaaS[10]などが挙げられる．

WebTTY，Anyterm ともに GNU General Public License のオープンソースで，本システムと同様にコマンドプロンプトを模した Web ブラウザで動作するシステムである．クライアント側で入力されたコマンドをサーバ側で実行し，出力された文字列をクライアント側へ返す．WebTTY は，ターミナルプロセスを HTML 要素経由で扱うため，Ajax，DHTML，C，シェルスクリプトなどを組み合わせた Web アプリケーションであり，Web ブラウザ経由でターミナルプロセスの操作を可能にする．

TextDrop は，Dropbox[11]内のファイルを編集することができるインストール不要の Web テキストエディタである．クロスブラウザに対応しており，ブラウザ上で動くのでスマートフォンやタブレット端末からでも使用可能である．

また，他の Web テキストエディタとして JavaScript でつくられている ACE が挙げられる．ブラウザ上で動作し，基本的なテキスト編集，シンタックスハイライト，行番号表示，簡易構文チェックなどの優れた機能を持ち，10 種類以上の言語に対応している．Vim や Emacs のキーバインディングの利用，検索や元に戻すなどのショートカットキーにも対応しており，様々なプロジェクトに使用されているテキストエディタである．

ACE を取り入れた開発環境として，Cloud9 IDE や Monaca IDE が挙げられる．Cloud9 IDE とは，Node.js[12]対応のサーバサイド JavaScript のための統合開発環境である．Node.js が動作するサーバの上に Web アプリケーションとして実装されているため，開発対象として Node.js を前提としている．Node.js とは，軽量で効率良く多くのリクエストを処理するネットワークアプリケーションの構築ができるプラットフォームである．サーバサイドのアプリケーションの実装言語として JavaScript を使用することで，クライアントサイドと同じ言語で開発でき，開発効率が上がる．コードエディタやファイル選択，シンタックスハイライトなどの機能があり，これらの機能は全て Web ブラウザから利用する．ソースコードのエラーチェックといった開発支援機能は実装されていないが，アプリケーションのデバッグ実行やステップ実行，変数の値の確認などの実行環境が用意されている．

Monaca とは，クラウドで提供するスマートフォン向けアプリ開発・運用環境プラットフォームである．Monaca では，全ての開発環境をサーバ側で動作させることで，利用者はブラウザさえあればスマートフォンやタブレット向けのアプリを作ることができる．現在，様々なスマートフォン向け OS が存在するが，現時点で Monaca が対応する OS は，iOS ならびに Android である．Monaca の大きな特徴は，スマートフォン向け HTML5 フレームワークの PhoneGap[13]が組み込まれており，PhoneGap を使用する際に必要な環境設定などは何も行わなくて良いことである．PhoneGap では，HTML5 と CSS，JavaScript を使用して，様々なプラットフォームに対応するアプリケーションの開発を行うことができる．また，Web と違い，カメラや GPS，加速度センサーなどの端末固有の機能にも対応するため，Web アプリ以上の柔軟性を持ったアプリの開発を行うことができる．Web サイトと同様，アプリの各画面は HTML で構成することができ，プログラムは JavaScript を使って記述できるので高度な処理も書くことができる．jQuery や Sencha Touch などの JavaScript ライブラリを活用して表現力の高いアプリを作ることも簡単である．現在はベータ版がリリースされており，サポートしているブラウザが限定されているため，Internet Explorer などでは実行することができない．

VNC の関連研究として，VM ware View というシステムがある．VM ware View は，PCoIP という画面転送プロトコルを用いて Web を介した仮想デスクトップ環境を提供する．画面の出力をパターン認識し，文章や画像，動画といった種類に分ける．ネットワーク回線速度の状況に応じ，低速であれば文章の転送を優先し，高速であれば画像データも鮮明に転送する．クライアント側に専用のソフトウェア，クライアントとサーバ間のネットワーク上に帯域監視用のデバイスが必要となる．

他にも，VNC を利用したサービスとして DEVaaS (Development as a Service)が挙げられる．DEVaaS では，統合開発環境をサービスとして提供することを提案している．KVM(Kernel-based Virtual Machine)と VMM(Virtual Machine Monitor)によって動作し，プロジェクト管理システムや BTS(Bug Tracking System)，VCS(Version Control System)，CI system(Continuous Integration System)，エディタ，開発システムといった様々なシステムをブラウザで操作することが可能である．

本研究室では，Web 上で開発を行いクライアント側で実行する環境(Server-side Development Client-side Execution

Environment)を提案している．スタンドアローンな環境でGUI プログラムを実行する場合，ユーザ毎に各種設定を行う必要がある．また，Web 上で使用することを考えた場合，VNC によってサーバの GUI 環境を利用できるが，サーバ側のメモリ負荷が多く，ユーザ毎に VNC リソースが必要となる．そこで，サーバ側で開発を行い，クライアント側で実行することによって，サーバ側の負荷を減らし，ユーザが簡単に GUI 環境を利用できるようにしている．

3. Web ベース開発環境

3.1 システムの構成

図 1 に本システムの構成を示す．ユーザは Web ブラウザを立ち上げることでシステムを利用できる．ブラウザ上のコマンドプロンプトにコマンド等を入力することで，Windows のコマンドプロンプトや UNIX のターミナルウィンドウと同様，コマンドの実行，プログラムのコンパイル・実行が行える．コマンドの処理はインターネットを通じてサーバが行うので，ユーザはインターネットに接続できる環境さえあればいつでもどこでも同一環境を利用することができる．

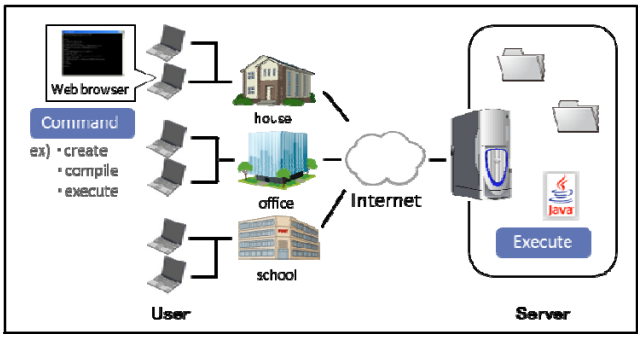

図 1　本システムの構成
Figure 1　Construction of Our System

3.2 開発・動作環境

本システムの開発環境及び動作環境を表 1 に示す．クライアント側を JavaScript，HTML，CSS，サーバ側を JavaServlet で実装し，ライブラリは prototype.js，YUI を用いた．

prototype.js は Ajax に対応した JavaScript ライブラリである．ブラウザ依存の部分を吸収するため，1 つの構文でクロスブラウザ対応が可能となる．本システムでは，サーバと HTTP 通信を行うため，prototype.js の Ajax.Request()を使用し，クロスブラウザ対応させた．また，ブラウザ上のコマンドプロンプト画面には YUI や CSS を用いることでデスクトップライクなユーザインタフェースの実装を行った．

表 1　開発・動作環境
Table1　Development and Operating Environment

	クライアント	サーバ
OS	WindowsXP	Windows 7
使用言語	HTML JavaScript	JavaServlet
使用ライブラリ	prototype.js YUI	
動作ブラウザ	Firefox 16.0.1	

3.3 コマンド実行方法

本システムでは，ブラウザ上のコマンドプロンプトを用いてCUIアプリケーションとGUIアプリケーションを実行することが可能である．通常のコマンドプロンプトと同様，コマンドの実行やプログラムのコンパイル・実行が行える．ここでは，CUI アプリケーションのバッチ処理手順を図 2 の番号に従って以下に示す．

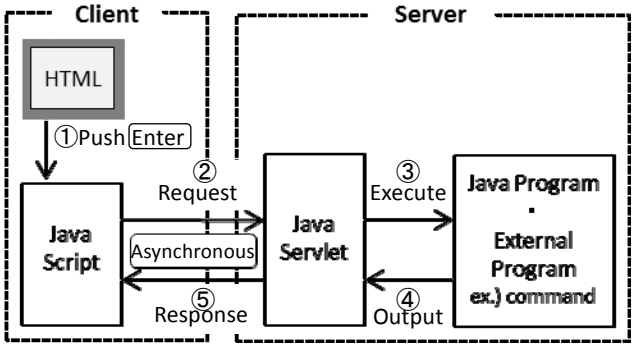

図 2　CUI アプリケーションのバッチ処理手順
Figure 2　Procedure of Batch Application

① ユーザがブラウザ上のコマンドプロンプトにコマンドを入力し Enter キーを押下すると，Enter キーに対する keydown イベントが発生．JavaScript で入力された文字列を解析し，コマンドの場合には「サーバと通信を必要としない」「サーバと通信を必要とする」の 2 つに分ける．前者の場合，クライアント側でのみ処理を行う．入力データの解析について図 3 に示す．

② ①の処理で「サーバと通信を必要とする」と判断した場合，コマンド文字列をリクエストとしてサーバ側に送信．

③ サーバ側でコマンド文字列を受信しコマンドを実行．

④ サーバプログラムはコマンドの実行結果として外部プログラムの出力結果を受け取る．

⑤ 外部プログラムの出力結果をクライアント側へレスポンスとして返信．

この実行方法は，クライアント側でユーザが入力したコマンド文字列をリクエストとして送信した後，サーバ側はプロセスでの外部プログラムの出力結果をレスポンスとして返している．そのため，1 度の通信でデータを複数回送受信することができず，バッチモードにのみ対応している．そこで，インタラクティブモードに対応させる方法について検討を行った．

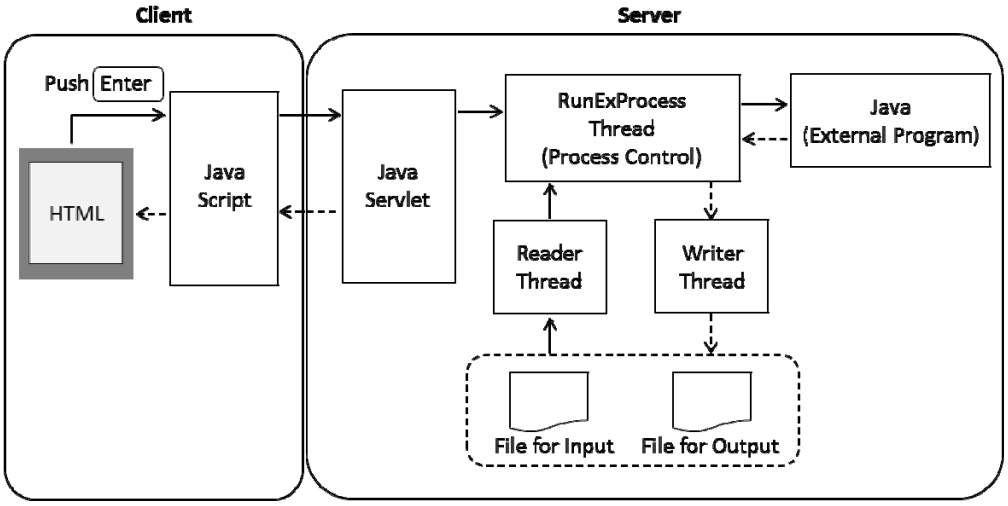

図 4　クライアント・サーバの構成

Figure 4　Construction of Client and Server

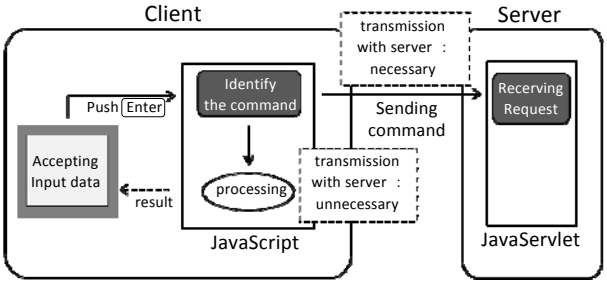

図 3　入力データの解析

Figure 3　Analysis of the Input Data

3.4 インタラクティブなシステムの構築

本システムでのインタラクティブモードの実現を目指すため, 入力用ファイルと出力用ファイルを追加することで, クライアント側とサーバ側でデータの送受信を行うようにした．クライアントとサーバ間の構成を図 4 に示し，以下にシステムの処理の流れについて説明する．なお，クライアント側での入力データの解析は 3.3①と同じなので省略する．

(1) クライアント側からリクエストが送られてきた場合，サーバ側は，プロセス管理用プログラムを呼び出し，外部プログラムを起動する．各スレッドの処理内容は以下の通りである．

- RunExProcess Thread

クライアント側から受け取った外部プログラムをプロセスとして起動･実行する. プロセス生成後に Writer Thread と Reader Thread の呼び出しを行う.

- Writer Thread

RunExProcess Thread からプロセスの出力を受け取り，出力用ファイルに 1 行ずつ書き込む.

- Reader Thread

入力用ファイルの中身を 1 行ずつ読み込み，プロセスの入力値として RunExProcess Thread へ渡す.

(2) クライアント側では，対応する出力用ファイルの中身を定期的に監視し，更新されていた場合，ブラウザ上のコマンドプロンプトにデータを出力する．また，外部プログラム実行中にユーザがブラウザ上のコマンドプロンプトに文字を入力した場合，文字列をサーバ側へ渡し，サーバ側で対応する PID フォルダ内の入力用ファイルにデータを書き込む.

(3) 外部プログラムが終了したら，サーバ側は入出力用ファイルを削除し，クライアント側はコマンドプロンプト画面の更新を終了する.

3.5 実行結果

図 5 に，本システムの実行結果を示す．ここでは，単純な出力プログラム(Hello.class)と，ユーザからの入力を受け取り，加算結果を表示するプログラム(adder.class)それぞれの実行例を示す．また，システム利用時の詳細な処理内容について以下に説明する.

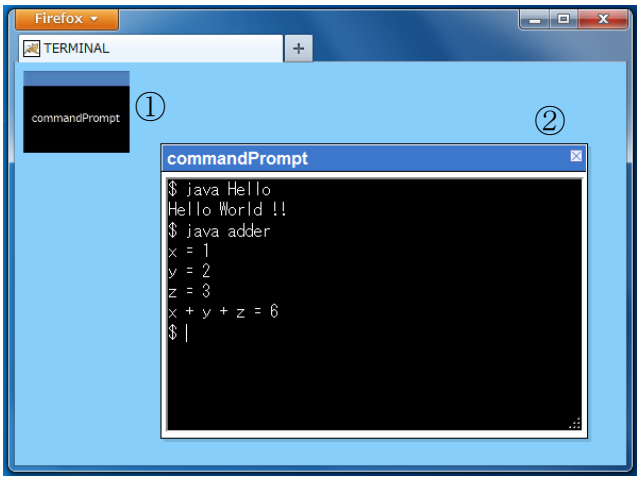

図 5　コマンド実行結果

Figure5　Result of Command Execution

(1) Web ブラウザ上にコマンドプロンプト用アイコン(図 5①)を表示する.

(2) ユーザがアイコンをクリックすることでコマンドプロンプト画面(図 5②)を出力し, ユーザからの入力を受け付ける. ユーザがコマンドを入力し Enter キーを押下すると, Enter キーに対する keydown イベントが発生する. 入力された文字列を解析し, コマンドの場合には「サーバと通信を必要としない」「サーバと通信を必要とする」の 2 つに分ける. 前者の場合には JavaScript 内でそのまま処理を行い, 実行結果をコマンドプロンプト画面に直接出力する. 後者の場合にはコマンド文字列をサーバ側へリクエストとして送信する. この時のコマンド実行方法と結果出力方法については(3)以下で説明する.

(3) (2)の処理で, ユーザが入力したコマンドをサーバ側へリクエストとして送信した場合, コマンドの処理はサーバ側で行う. この時にサーバ側で行う処理を図 6 に示す.

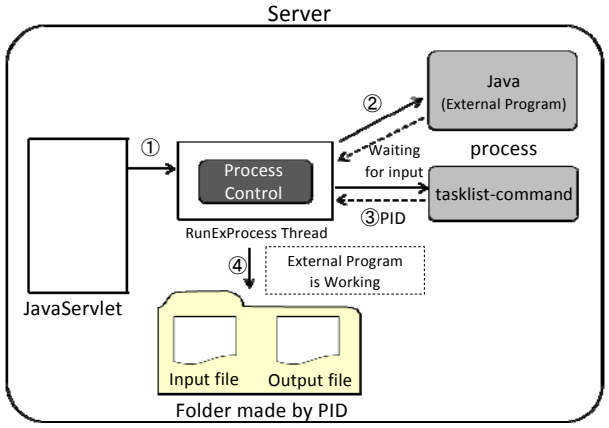

図 6 サーバ側でのコマンド処理ルーチン

Figure 6 Command Processing Routine on the Server Side

① プロセスの実行・管理を行う RunExProcess Thread を呼び出す.

② RunExProcess Thread では, リクエストとして受け取ったコマンドをプロセスにて実行する.

③ tasklist コマンドを実行することで, プロセス実行した外部プログラムの PID を取得する. tasklist とは Windows コマンドであり, 実行しているタスクの内容をアプリケーションとサービスの一覧, PID とともに表示する.

④ 実行した外部プログラムの PID を取得した後, その PID に対応するフォルダと入出力用ファイルを生成する. サーバ側でコマンド実行処理を行う場合には, この PID フォルダを用いてクライアント側とデータの送受信をする. 外部プログラム実行プロセスが終了した場合, 対応する PID フォルダと入出力用ファイルを削除する. なお, データの送受信に関する詳細については(4)で説明する.

(4) サーバ側でコマンド実行処理を行う場合には, (3)で生成した PID フォルダ内の入出力用ファイルを用いてクライアント側とデータの送受信をする. クライアントとサーバ間での入出力データの流れを図 7 に示す. なお, 斜線付き矢印は, 入出力データの流れを示している.

RunExProcess Thread では, まず初めにプロセス状態を確認し, プロセス生成後に Reader Thread と Writer Thread の呼び出しを行う. クライアント側では, レスポンスとして PID を受け取り, 対応する PID フォルダ内の出力用ファイルデータの確認を行い, 新たな出力データがある場合にはブラウザ上のコマンドプロンプト画面へデータを出力する.

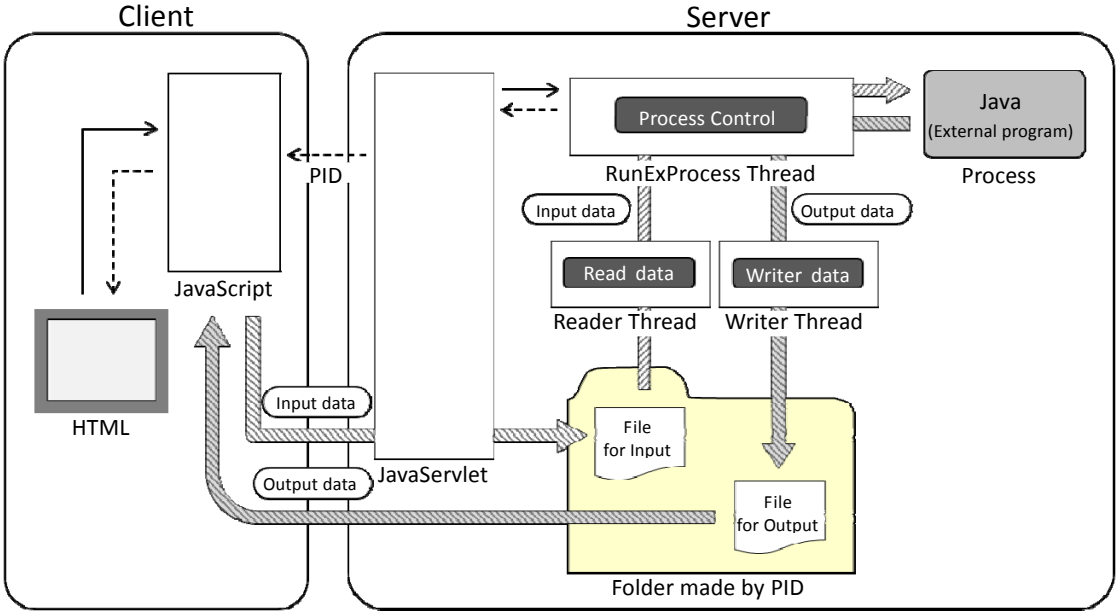

図 7 クライアント・サーバ間での入出力データの流れ

Figure 7 Data Flow between Client and Server

また，プロセスが入力待ち状態の際に，ユーザがブラウザ上のコマンドプロンプト画面に値を入力して Enter キーを押下すると，Enter キーに対する keydown イベントが発生し，ユーザが入力した値と，現在実行している外部プログラムに対応する PID をサーバ側へリクエストとして送信する．サーバ側は，対応する PID フォルダ内の入力用ファイルに受け取ったデータを書き込む．
(5) (4)の処理によって，複数の入力値が必要なプログラムにも対応させた．また，クライアント側では外部プログラムが終了したら出力用ファイルの監視を止め，ユーザから新たなコマンドの入力を受け付ける．

4. 評価

本システムの評価を行うため, PC 端末からシステムへアクセスし，サーバ側での PID フォルダ生成時間とコマンドに対する応答時間についてそれぞれ計測を行った．なお，コマンドに対する応答時間の計測には Firefox のアドオンである Firebug[14]を用いた．

まず，サーバ側での PID フォルダ生成時間について 10 回計測を行った結果，平均時間は 407.7ms，中央値は 257.9ms となった．次に，コマンドに対する応答時間について 10 回計測を行った結果，平均時間は 470ms，中央値は 309ms となった．また，コマンドに対する応答時間の最小時間は 228ms，最大時間は 1080ms となり，大きな差が生じた．この時のコマンドに対する応答時間と PID フォルダ生成時間の占める割合はほぼ一定であったため，サーバ側のフォルダ生成処理によってコマンドに対する応答時間に影響が出たことがわかった．

5. おわりに

本研究では Web ベースシステムの実装を行っており，CUI と GUI プログラムの両方に対応しているが，CUI についてはバッチ処理プログラムのみしか実行できなかった．そこで，より柔軟な入力処理を行うため，ユーザからの入力によってリアルタイム処理を行うインタラクティブ入出力に対応させる方法について検討した．サーバ側でプロセスによって外部プログラムを実行した場合，プロセスの入力待ち状態を検出することはできないので，スレッドとして読み込み処理と書き込み処理を行うようにした．また，PID 毎に専用の入出力用ファイルを生成し，その入出力用ファイルによってクライアント・サーバ間でデータの送受信を行うことでインタラクティブな柔軟な入力に対応させた．

今後の課題として，応答時間はサーバの負荷によっても変化するため，複数人でシステムを使用した場合のパフォーマンスの評価についても行う必要がある. また，HTML5 を用いた実装方法についても検討したい．

参考文献

1) 小寺勇司, 柳澤秀明: Web ベース Java プログラミング環境の開発，平成 22 年度電気・情報関連学会中国支部連合大会論文集, pp.419-420(2010-10)
2) 佐内修平, 柳澤秀明: Web ベースプログラミング環境の開発, 情報処理学会研究報告マルチメディア通信と分散処理(DPS), 2012-DPS-150(22), 1-7, 2012-02-22
3) WebTTY
http://testape.com/webtty_sample.html
4) Anyterm
http://anyterm.org/
5) TextDrop
https://www.textdropapp.com/home/Home
6) ACE
http://ace.ajax.org/#nav=about
7) Cloud9 IDE
https://c9.io/
8) Monaca Beta
http://monaca.mobi/?lang=ja
9) VM ware View
http://www.vmware.com/jp/products/desktop_virtualization/view/overview
10) Katsuyoshi Matsumoto, Shinichiro Kibe, Minoru Uehara, Hideki Mori: Design of Development as a Service in the Cloud, Proc.of the 1st International Workshop on Renewable Computing Systems(WReCS-2012), CD-ROM, pp.815-819(2012)
11) Dropbox
https://www.dropbox.com/
12) Node.js
http://nodejs.org/
13) PhoneGap
http://phonegap.com/
14) Firebug
https://addons.mozilla.org/ja/firefox/addon/firebug/