科学研究費助成事業　　研究成果報告書

平成　２６　年　　６　月　　９　日現在

機関番号：　１２６１２
研究種目：　基盤研究(C)
研究期間：　2011～2013
課題番号：　２３５０００３８

研究課題名（和文）実用的ウェブアプリケーション開発を支援するサーバサイドＪａｖａＳｃｒｉｐｔ処理系

研究課題名（英文）A Server-side JavaScript System for Developing Practical Web Applications

研究代表者
岩崎　英哉（Iwasaki, Hideya）

電気通信大学・情報理工学（系）研究科・教授

研究者番号：９０２０３３７２

交付決定額（研究期間全体）：（直接経費）　　4,000,000円、（間接経費）　　1,200,000円

研究成果の概要（和文）：本研究は，サーバ側で実用的な性能で動作する サーバサイドJavaScript処理系を開発することにより，煩雑な Web アプリケーション開発のコストを大きく低減させることを目指し，以下の成果を得た．(1) プログラムの実行情報を利用した最適化を行い実行速度を向上させるJavaScript処理系を構築した．(2) JavaScriptプログラムとC言語で記述されたプログラムとの連携を可能とする機構を構築した．(3) イベント駆動方式サーバのための並列JavaScript処理系を設計し実装した．(4) JavaScriptプログラムの安全な実行の基礎となる型システムを構築した．

研究成果の概要（英文）：This research aims to develop an efficient server-side JavaScript engine that is expected to reduce the development cost of web applications. The obtained results can be summarized as follows. (1) We have developed a JavaScript virtual machine that optimizes the program execution on the basis of run-time information. (2) We have developed a foreign function interface that enables the programmer to call C functions from a JavaScript program. (3) We have extended the JavaScript virtual machine described in (1) so that it can run in parallel as event-driven servers. (4) We have developed a gradual type system for a subset of JavaScript programs to be a base of their safe execution.

研究分野：　総合領域

科研費の分科・細目：　情報学・ソフトウェア

キーワード：　JavaScrip　仮想機械　実行時最適化　Just-in-timeコンパイラ　外部関数インタフェース　イベント駆動方式サーバ　漸進的型システム

１．研究開始当初の背景

近年 Web アプリケーションと呼ばれる Web ページが広く利用されるようになった．その一方で，Web アプリケーションの開発は，非常に煩雑であることが知られている．その大きな理由として，サーバ側とクライアント側の両方のプログラムを，異なる言語で記述する点が挙げられる．クライアント側のプログラムはブラウザ内で動作するため，その記述言語は JavaScript とするのが自然であり，最も一般的である．一方，サーバ側のプログラムは，Perl，Java など多様なプログラミング言語が利用されている．このように，サーバとクライアントで異なるプログラミング言語を用いて記述するため，そもそも異なる文法を持つ複数の言語によるプログラミングを強いられるのに加え，言語自身の記述力，データ内部表現，セマンティクスの違いにも気を配らなければならず，アプリケーションの効率的な開発の足枷になっている．

本研究は，サーバのプログラムを JavaScript で記述し，サーバ側で作動する JavaScript 処理系（以後「サーバサイド JavaScript」と呼ぶ）で実行できれば，言語の違いに起因する上の問題点は解消され得ること，及び，現状では性能の良いデファクトスタンダードであるサーバサイド JavaScript の処理系が存在しない，という点に注目した．さらに，研究開始時の背景としては，Node.js と呼ばれる，イベント駆動による通信を支援するライブラリのように，サーバサイド JavaScript に関する仕様を定めようとする動きがあること，および，JavaScript の標準規格である ECMAScript バージョン 5 が 2009 年 12 月に標準化されたことといった点をあげることもできる．

２．研究の目的

本研究は，上で述べた背景のもと，サーバ側で実用的な性能で動作する サーバサイド JavaScript 処理系を研究開発することにより，煩雑とされている Web アプリケーション開発のコストを大きく低減させ，広く一般の利用に供することを目的とする．この目的を達成するため，代表者が行ってきたプログラミング言語処理系，特に記号処理言語・関数型言語に関する研究と，二人の分担者が行ってきた，プログラミング言語処理系の実行時システムの研究，および，型理論やプログラム変換等のプログラミング言語に関する理論的研究を有機的に結びつけ，JavaScript プログラムから仮想機械命令へのコンパイラ，仮想機械命令から機械語への Just-in-time コンパイラ，自動的メモリ管理を有する仮想機械等の機構を持つ，サーバサイド JavaScript システムの構築を目指す．

本研究で目指す「実用的」とは，具体的には一般的なアプリケーションの記述と運用に耐えうることを意味する．そのために，次の 3 点を満足する JavaScript 処理系を構築することを目標とした．

高い性能: プログラムが高速に動作するという時間的な側面の性能と，巨大なアプリケーションの動作が可能であるという空間的な側面の性能を両立させ，様々なアプリケーションに適用可能なシステムとする．そのために，効率の良い仮想機械命令列を生成するコンパイラ，仮想機械命令列から実行時にネイティヴコードを生成する Just-in-time コンパイラなどを開発する．

高い拡張性: サーバサイドで動作するプログラムは，C 言語等の JavaScript 以外の言語で書かれたプログラマ自作ライブラリなどを利用するケースが多いと考えられる．したがって，アプリケーション開発者にとってやさしく，本処理系にとってもオーバーヘッドの少ないような，ネイティヴライブラリを呼び出すインタフェースを用意し，高い拡張性を提供する．

高い安全性: Web アプリケーションのサーバ用途では，サーバプログラムの安全性が重要な関心事である．そこで，型検査等の静的解析機構を利用し，たとえば，プログラムがある種のエラーを発生しないこと，機密データがサーバ・クライアント間に流れることがないこと等の，安全性に関する性質を静的に保証する．

３．研究の方法

本研究で開発する JavaScript 処理系は，仮想機械の構成をとる．すなわち，JavaScript プログラムから中間言語である仮想機械命令列へのコンパイラと，その命令列を実行する仮想機械を開発する．このような構成は，処理系の柔軟性を増し，目標を逐次的に定めた段階的なシステム設計・開発を可能にするという大きなメリットがある．

本研究の具体的な進行手順は，まず JavaScript の中心的な機能を抽出したサブセット（以下 Tiny JavaScript と呼ぶ）を定め，Tiny JavaScript に対するコンパイラと仮想機械（以下 SSJSVM と呼ぶ）を実現する．この仮想機械はレジスタマシンの構成をとった上で，仮想機械命令セットを規定する．これらの作業を通して，処理系実装上の問題点等を明らかにする．さらにこの経験をもとに，SSJSVM をより洗練し性能を向上させる．具体的には，実行時情報を有効に活用した各種最適化の実装を行う．

上と並行して，JavaScript プログラムに対する型検査などの静的解析機構を開発する．型検査については漸次的型付けを考える．

４．研究成果

本研究で得られた成果を，以下にまとめる．

(1) JavaScript 仮想機械 SSJSVM の開発を行い，実行時最適化の効果を確認した．SSJSVM の開発にあたり，JavaScript のサブセットである Tiny JavaScript を規定した．このサブセットは，eval 文，with 文のような，仮想機械およびコンパイラの実装を複雑化する要因となるものをサポートしないこととした．このような制限を設けても，実用的なサーバプログラムの記述の大きな障害にはならないと判断した．

まずはじめに，Tiny JavaScript のプログラムを SSJSVM の仮想機械命令列にコンパイルするコンパイラを開発した．このコンパイラは以降の研究においても利用され，本研究における基本的な役割を果たしている．

JavaScript のプログラムは，実行するまで変数の型が定まらず，またプログラムの実行中に変数の型が変わることがあるので，実行時情報を利用した最適化が効果的である．そのような最適化に Quickening がある．

Quickening とは，プログラムの実行中にプログラム中の仮想機械命令を特殊化された別の仮想機械命令に置換することにより，実行時性能を上げる手法である．ここでは，型変換を伴う可能性のある演算命令と，大域変数へアクセスする命令を Quickening の対象とした．ここで，置換後の特殊化された命令は，あらかじめ用意しておいた．

演算命令については，SSJSVM において Quickening の対象とする各演算命令に対して，前回その命令を実行した際のオペランドの型の組合せと，その組み合せが何回連続して現れたかを記録するようにした．その連続回数があらかじめ定められた閾値を超えたら，その演算命令を連続して出現した型に特殊化した命令に置換する．

大域変数へのアクセス命令については，以下のように命令の置換を行う．SSJSVM では，大域変数へのアクセスはグローバルオブジェクトと呼ばれる特別なオブジェクトのプロパティへのアクセスとして実現されている．これはグローバルオブジェクトが持つハッシュ表に対して，大域変数名（プロパティ名）のハッシュ値を添字としてアクセスする．このハッシュ値はプログラムに実行中に変化することがないので，大域変数へのアクセス命令を最初に実行した時に，その大域変数が格納されているハッシュ表における場所に直接アクセスするように特殊化した命令に置換する．

これらの実行時最適化の効果を，いくつかのベンチマークプログラムを用いて確認した．この成果は，日本ソフトウェア科学会第 15 回プログラミングおよびプログラミング言語ワークショップ（PPL 2013）において論文が採択され，登壇発表を行った．

(2) サーバサイド JavaScript 処理系における Just-in-time (以下 JIT)コンパイラを，前項で開発した SSJSVM の上に実装し評価した．(1) で述べたように，JavaScript プログラムは実行するまで変数の型が定まらず，型によってプログラムの振る舞いが変化する．そのため，JavaScript プログラムに対しては，実行時情報を用いた最適化が効果的であると考えられる．そのような最適化の中で，この研究では実行時の動的な情報を用いて，頻繁に実行されるプログラムコード (SSJSVM の仮想機械命令列）の一部をマシンコードに変換し，そのコードを複数回実行することにより実行時間の短縮をはかる JIT コンパイラに注目した．

一般的に，サーバサイドではプログラムの長期的な運用を前提として考えることができる．そのため，複数回実行される可能性のあるコードは，長期的な視野では多くの回数実行されると考えられる．これより，最適化にある程度の時間を要しても，実行の早い段階で十分な最適化を施したマシンコードを生成するという方針をとった．JIT コンパイルの対象とするコード部分（ホットスポット）としては，1 回でも実行に及んだユーザ定義関数とした．

この研究での JIT コンパイラ機構では，対象のプログラムを直接マシンコードに変換するのではなく，コンパイラ基盤として広く用いられている LLVM の中間表現である LLVM-IR に変換する．これにより，LLVM の提供する最適化機構を間接的に利用し，アークテクチャに合わせたマシンコードを生成することが可能となる．

本処理系の実装は，SSJSVM による仮想機械命令列実行のモニタリング機構，仮想機械命令列から LLVM による十分な最適化を可能とするための型推論器，前項の結果から LLVM-IR を生成する LLVM-IR 生成器からなる．これらの機能を，前項(1)で開発した SSJSVM を拡張して実装した．

提案した JIT コンパイラの効果を確認するため，数値計算を行うプログラムをはじめとして，いくつかのベンチマークプログラムを用意し，既存の JavaScript 処理系である V8 と SFX を実行時間を比較した．その結果，長期間の運用を前提とした場合，最適化を含むマシンコード生成のコストはかかるものの，提案機構の効果が確認できた．

この成果は，日本ソフトウェア科学会第 16 回プログラミングおよびプログラミング言語ワークショップ（PPL 2014）において論文が採択され，登壇発表を行った．

(3) JavaScript 処理系に対する外部関数インタフェースを設計し実装した．サーバサイドにおける開発では，実行効率の向上，既存ライブラリの利用，オペレーティングシステム固有機能の利用などの理由から，C など他

言語で記述されたプログラムと連携を行いたいという要求がある．このような要求は，外部関数インタフェース（Foreign Function Interface，以下 FFI）と呼ばれる仕組みを提供して解決するのが一般的である．

この研究では，C 言語で定義された関数を JavaScript から呼び出すための FFI を設計し，SSJSVM 上に実装した．本 FFI の特徴は，次の通りである．1) JavaScript と C のソースコードのみで完結しており，他ツールなどを必要としない；2) C の関数は JavaScript 側から容易に呼び出すことが可能である；3) C の関数から JavaScript のグローバル環境を参照できる；4) C 関数呼出し時に実引数の型や個数の自動チェック機構がある．

本 FFI を既存の JavaScript 処理系である Node.js の FFI と比較すると，C 側から利用できる JavaScript の機能が豊富であること，実引数の自動チェックなど既存 FFI には無い機能を提供していること，などで優位性がある．被験者による評価結果でも，デバッグのしやすさに関しては改善の余地があるものの，書きやすさに関して高い評価を得ることができた．

この成果は，日本ソフトウェア科学会第 15 回プログラミングおよびプログラミング言語ワークショップ（PPL 2013）においてポスター発表を行った．

(4) イベント駆動方式サーバのための並列 JavaScript 処理系を設計し実装した．JavaScript では，サーバ記述に必要なネットワーク通信などのための API が仕様に含まれていないため，サーバサイド向けの JavaScript 処理系は独自の API を提供する必要がある．また，並列処理の仕様も定められていないため，サーバを並列動作させるためには，処理系が独自に提供する並列処理機構が必要である．しかし，広く利用可能な並列処理機構はクライアントサイドでの利用を想定して設計されているため，サーバの記述には必ずしも適していない．そのため，並列動作するサーバを記述するためには，サーバ記述のための API とサーバ向けとして十分に設計された並列処理機構が必要である．

この研究では，イベント駆動方式に基づくサーバに着目して，この方式のサーバを自動的に並列実行するための機構を設計し，SSJS VM 上に実装した．本機構では，サーバは次のようにして記述する．1) サーバが処理すべきイベントそれぞれに対して，イベントハンドラを定義する；2) 必要に応じてサーバ実行に関するパラメータを設定し，イベントループを開始する．イベントハンドラを登録した後に，サーバオブジェクトが持つ起動用メソッドを呼び出すことで，サーバの実行が開始されイベント処理が行われる．イベントループは API の内部に隠蔽されているため，JavaScript プログラムに明示的に現れることはない．並列動作は，イベントハンドラを複数のスレッドで並列実行することで実現するという方針をとった．

以上のように，並列実行の制御を処理系が提供する API の内部で自動的に行うことにより，JavaScript プログラム中の並列実行の制御を陽に記述することによるサーバ記述の複雑化を避けながら，サーバの並列実行を実現することができた．

この成果は，日本ソフトウェア科学会第 16 回プログラミングおよびプログラミング言語ワークショップ（PPL 2014）においてポスター発表を行った．

(5) JavaScript プログラムに対する漸進的な型システムを定義した．漸進的な型システムにおいては，プログラマは静的に型検査を行う部分を明示的に指定する．処理系はその指定された部分に関して静的に型検査を行い，残りの部分については動的に型検査を行う．これにより，プログラムに対して徐々に型を付け，静的に型検査を行う部分を増やしていくことができる．静的に型が付けれられた部分に関しては，実行時に型エラーが起こらないことが保証される．

JavaScript に対する漸進的な型システムを定義するために，JavaScript のサブセットを定め，それを対象とした．さらにこの JavaScript のサブセットを，破壊的代入（副作用）を持たない言語（以下 GJS と呼ぶ）に変換した．GJS では，型が宣言されておらず動的に型検査を行う変数に対して，型「?」を与える．この GJS に対する漸進的な型システムの型付け規則を定義した．

さらに GJS のプログラムに対して型のキャストを挿入し，プログラムを変換した．この変換先の（キャストが挿入された）言語を IJS と呼ぶ．GJS の式が GJS の型付け規則で型を付けることができれば，IJS への変換は必ず存在する．キャストは，静的に型が付けられた部分と動的に型が付けられた部分の境界で挿入し，実行時の型検査はこの境界で行われる．続いて IJS に対する small step 意味論を定義し，IJS の健全性，さらに GJS の健全性を証明した．

本研究の成果は，研究代表者が指導教員となっている修士課程の学生の修士論文としてまとめられている．

(6) プログラム実行時のメモリ管理に関する研究として，マークコンパクトごみ集めにおいて，ヒープをスキャンする新しい方法を提案した．ごみ集め（Garbage Collection，以下 GC）とは，ヒープ上の使われることがなくなったオブジェクトによって占められていた領域（ごみ)を回収し再利用を可能とするための機構である．ここで，ごみでないオブジェクトを「生きているオブジェクト」と呼ぶ．「マークコンパクトごみ集め」とは，生きているオブジェクトをヒープの端に詰

めるコンパクションと呼ばれる操作を行い，ヒープ領域の断片化を解消する．断片化の解消は，サーバサイドにおけるプログラムなど，長期間の運用が想定されるプログラムにとって極めて有用であり，サーバサイド JavaScript のような処理系に求められる機能である．

広く使われているコンパクションのアルゴリズムでは，ヒープを端から順に走査し，生きているオブジェクトを探す．この走査をスキャンと呼ぶ．ヒープ領域が大きくなるとヒープのスキャンに時間がかかり，その分プログラム実行のオーバーヘッドになってしまう．このオーバーヘッドを軽減するために，ヒープとは独立にオブジェクトの生死情報を記録したビットマップを用意し，スキャンの対象をヒープそのものにするかビットマップにするかを動的に適切に切り替える手法を提案した．

提案する手法を研究用の Java 仮想機械処理系である Jikes RVM に実装し評価したところ，本手法のスキャン方法が多くの場合で既存のスキャン方法より高速であることがわかった．

本手法は現状においては SSJSVM 上に実装はしていないが，SSJSVM においても有用であることが予想される．

この研究の成果は，メモリ管理に関する国際シンポジウム (ISMM 2013)において論文が採択され，登壇発表を行った．

５．主な発表論文等

〔雑誌論文〕（計 3 件）

① 高田祥，鵜川始陽，中野圭介，岩崎英哉：JavaScript 処理系における Quickening の効果，第 15 回プログラミングおよびプログラミング言語ワークショップ（PPL 2013）論文集，2013，査読有．

② Kazuya Morikawa, Tomoharu Ugawa, Hideya Iwasaki: Adaptive Scanning Reduces Sweep Time for the Lisp2 Mark-Compact Garbage Collector, Proc. 2013 International Symposium on Memory Management (ISMM 2013), ACM Press, pp.15-26, 2013, DOI:10.1145/2464157.2466480, 査読有．

③ 漆原明博，岩崎英哉，鵜川始陽：サーバサイド向け JavaScript 処理系における Just In Time コンパイラの実装，第 16 回プログラミングおよびプログラミング言語ワークショップ（PPL 2014）論文集，2014，査読有．

〔学会発表〕（計 5 件）

① 田村知博，中野圭介，鵜川始陽，岩崎英哉：JavaScript におけるプログラム変換の効果，夏のプログラミング・シンポジウム 2011「プログラミング言語，作る人，使う人」報告集，2011 年 9 月 2 日，佐賀県唐津市，pp.19-26，2011．

② 高田祥，鵜川始陽，中野圭介，岩崎英哉：サーバサイド JavaScript 処理系の開発について，第 14 回プログラミングおよびプログラミング言語ワークショップ（PPL 2012），2012 年 3 月 9 日，和歌山県白浜町．

③ 森川和哉，鵜川始陽，岩崎英哉：ビットマップマーキングを利用したマークコンパクトごみ集めの Jikes RVM への実装，情報処理学会第 89 回プログラミング研究会，2012 年 6 月 21 日，北海道小樽市．

④ 谷村明，岩崎英哉，中野圭介，鵜川始陽：JavaScript 処理系に対する外部関数インタフェースの設計及び実装，第 15 回プログラミングおよびプログラミング言語ワークショップ（PPL 2013），2013 年 3 月 4 日，福島県会津若松市．

⑤ 藤井亮太，岩崎英哉：イベント駆動式サーバのための並列 JavaScript 処理系の実装，第 16 回プログラミングおよびプログラミング言語ワークショップ（PPL 2014），2014 年 3 月 6 日，熊本県阿蘇市．

〔その他〕
ホームページ
http://ipl-www.cs.uec.ac.jp/~iwasaki/SSJS

６．研究組織

(1)研究代表者
岩崎 英哉（IWASAKI, Hideya）
電気通信大学・大学院情報理工学研究科・教授
研究者番号：90203372

(2)研究分担者
中野 圭介（NAKANO, Keisuke）
電気通信大学・大学院情報理工学研究科・准教授
研究者番号：30505839

(3)研究分担者
鵜川 始陽（UGAWA, Tomoharu）
電気通信大学・大学院情報理工学研究科・助教
研究者番号：50423017