5L-10

JavaScript におけるマルチエージェントシステムの実現とその応用

伊藤 正都 † 大囿 忠親 ‡ 新谷 虎松 ‡
名古屋工業大学知能情報システム学科 † 名古屋工業大学大学院工学研究科情報工学専攻 ‡
e-mail: {itomasa, ozono, tora}@ics.nitech.ac.jp

1 はじめに

本稿では，Web ブラウザ上で動作する JavaScript を用いてのマルチスレッドの実現，およびマルチエージェントシステムについて述べる．近年，Web において Web ページ閲覧者のユーザビリティーを向上させる為に，Web ブラウザ上で JavaScript を用いたダイナッミックな表現手法，およびリソース消費の大きい処理をする機会が増大している．しかしながら，Web ブラウザ上で動作する JavaScript プログラムは基本的にシングルタスクで動作しており，同時に，かつ数多くのダイナミックな表現処理を容易に行うことは困難である．また，リソース消費が大きい JavaScript プログラムの実行時には，プログラムの処理に計算機のリソースを大きく消費する為，Web ブラウザがユーザからの操作要求を満たせない場合がある．以上のように，JavaScript プログラムにおいて，同時に複数のダイナッミックな処理，およびリソース消費の大きい処理を容易，かつ Web を閲覧しているユーザの妨げとなること無く行うことは困難である．これらの問題を解決するにあたり 2 つの提案を行う．

1 つ目の問題解決手法として，JavaScript プログラムでのマルチスレッド機能の提供を提案する．JavaScript プログラムでのマルチスレッド機能の提供とは，プログラマに対して JavaScript プログラムでの擬似的なマルチスレッド実行環境を提供することである．しかしながら，JavaScript は基本的にはシングルタスク言語なので，擬似的にマルチスレッド機能を提供する必要がある．

2 つ目の問題解決手法として，JavaScript マルチエージェントシステムを提案する．これは，1 つ目の問題解決提案である JavaScript プログラムでのマルチスレッド機能を用いたものである．マルチエージェントシステムを用いることで，エージェントを複数利用したシステムの作成が可能となる．マルチエージェントシステムにおいて，プログラマが要求する処理をエージェント毎の機能として実装することで，擬似的ながら同時に複数の処理するように見せかけることを，容易に行うことが可能となる．

本稿では，JavaScript 以外を用いない JavaScript プログラムのマルチスレッド化手法を述べる．また，JavaScript マルチエージェントシステムの試作について述べる．

2 JavaScript におけるマルチスレッド化手法

本節では，JavaScript プログラムのマルチスレッドを実現する手法例を紹介し，伊藤 [2] の手法を改良した JavaScript マルチスレッド手法を説明する．

2.1 JavaScript マルチスレッドの既存手法

- Rhino[1]
  Rhino は，Java で実装された JavaScript 処理系である．Rhino，XPConnect[2]および XPCOM[3]を用いることにより JavaScript と Java を相互に利用可能となる．Java のスレッド機能を用いマルチスレッドプログラミングが可能となる．Rhino を用いた手法では JavaScript のみでマルチスレッド機能を実現できない．また，JavaScript プログラマが Java を新たに習得する必要があり，プログラマの多大な負担となる．
- 本研究の提案手法
  本研究では，JavaScript プログラミングに対して，次の制限をすることで JavaScript のみでのマルチスレッドプログラミングを可能としている．
  - スレッド内では別のスレッドを呼び出さない
  - スレッドを切り替える時間内に終了しない処理は行わない
  - スレッド間で使用する変数はグローバル変数と相互に変換可能である
  - スレッド内で使用する変数は使用前にグローバルな変数リストに格納する

  これらの制限を行うことで，JavaScript プログラムでのコーペラティブマルチタスクスレッドモデルを提案している．

2.2 提案手法

本提案手法の特筆すべき点は，容易なプログラミングを可能とし，またプログラミングの制限を緩くした点である．スレッド毎の動作をオブジェクトとして定義することにより，容易なスレッドプログラミング環境を提供する．以下にスレッドプログラミングの例を示す．

```
1 var thread = new Client(threadChannel, thread= function(){
2     alert("This is thread test!");    実際に処理をする内容
3     thread.start();
4     }
5 );
6 thread.start();
```

ここで各行左側の番号は説明のために付けた行番号である．1 行目から 5 行目までがスレッドの宣言となり，6 行目がスレッドの実行を定義している．1 行目でスレッドオブジェクトを作成し，2 行目において実際に処理をする内容を記述する．3 行目は再帰的にスレッドを実行する際に必要な宣言である．

3 JavaScript マルチエージェントシステム

JavaScript を用いたエージェントシステムとしては MiSpider[1]，airWeb[4]等が挙げられる．しかしながら，本研究では第 2 節で示した JavaScript のマルチスレッド機能

Building Multi-Agents System based on JavaScript Programming
Masato ITO, Tadachika OZONO, and Toramatsu SHINTANI
Dept. of Intelligence and Computer Science, Nagoya Institute of Technology, Gokiso, Showa-ku, Nagoya, 466-8555 JAPAN

[1] http://www.mozilla-japan.org/rhino/
[2] http://www.mozilla-japan.org/scriptable/
[3] http://www.mozilla-japan.org/projects/xpcom/
[4] http://www.airclub.org/

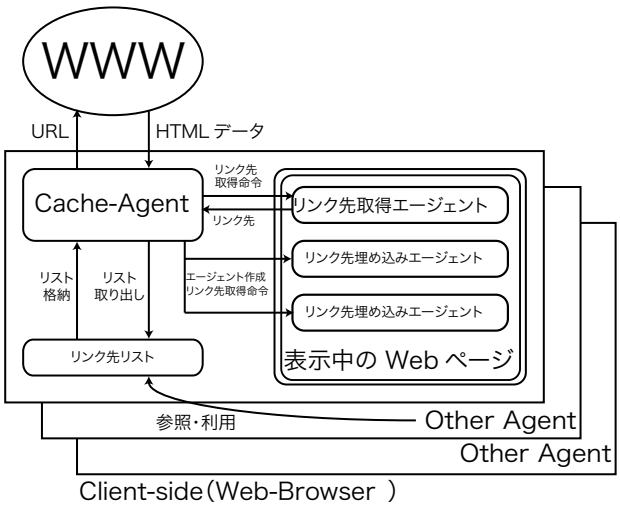

図 1: キャッシュエージェントシステム構成図

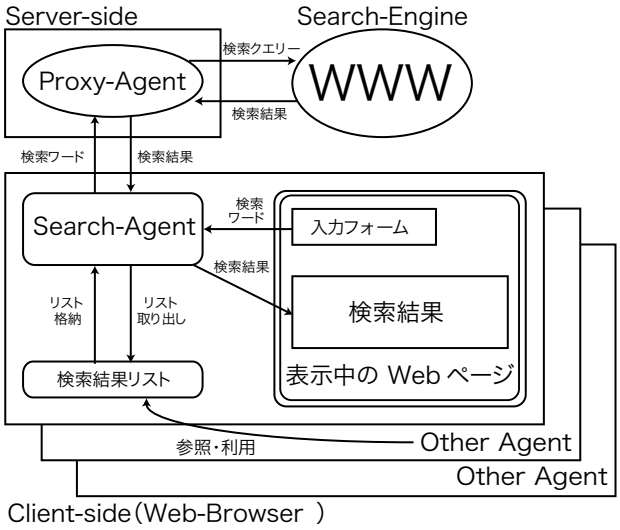

図 2: 検索エージェントシステム構成図

の提供により，マルチエージェントシステムが作成可能となる．1 つのスレッドを 1 エージェントとする場合には容易にエージェントの作成が可能である．開発された，1 スレッドからなるエージェントは，イベント発生時，または Web ページの読み込み完了時に実行可能な状態にすることで，スレッドとして容易に実行可能である．このことは，容易にエージェントを作成，実行する上で重要であると考えられる．エージェント間での通信が必要な場合には，メッセージパッシング能力を必要とする．本研究でのマルチエージェントシステムでは JavaScript におけるスレッドオブジェクトの相互参照によりメッセージパッシング機能を実現している．プログラマはメッセージパッシングが必要な場合，エージェントとして利用されるスレッドオブジェクトに相互参照可能な変数オブジェクトを持つことで，メッセージパッシング機能の代替として利用することが可能となる．

次に，本研究での JavaScript マルチスレッド機能を用いて作成したエージェント例を示す．これらは容易に同一 Web ページ上に存在が可能である．

3.1 キャッシュエージェント

キャッシュエージェントは，ユーザが閲覧中の Web ページからリンクされている Web ページへ遷移を行う前に，ユーザが閲覧中の Web ページからのリンク先を取得するエージェントである．キャッシュエージェントを用いて，リンク先のデータを先読みすることにより，ユーザは Web ページの遷移を高速に行うことが可能となる．これは，Web ブラウザが一度読み込んだ Web ページを保存し，再度同じ Web ページが呼び出された際には，既に保存してあるページを表示するためである．

キャッシュエージェントは，ユーザが閲覧するページに設置されることによって，Web ページの制作者の意図においてリンク先をユーザの Web ページ遷移以前に取得させることが可能となる．また，リンク先を取得する際には，ユーザが閲覧している Web ページ上に存在し，かつユーザのモニタに現在表示しているリンク先のみを取得することも可能である．キャッシュエージェントは次の手順を踏む．(i)Web ブラウザ上で設置されたキャッシュエージェントがリンク先取得エージェントを呼び出す．(ii) リンク先取得エージェントがリンク先の URL をリンク先リストに格納する．(iii) リンク先埋め込みエージェントは取得されたリンク先リストを用いて，Web ブラウザ上に表示させずに読み込む実際のキャッシュエージェントのシステム構成図は図 1 の通りである．

3.2 検索エージェント

検索エージェントは，ユーザが閲覧中の Web ページに Web 検索エンジン設置するするエージェントである．検索エージェントを用いることで，ユーザが閲覧中の Web ページに検索エンジンで得られた結果を表示することが可能となる．検索エージェントは次の手順を踏む．(i)Web ブラウザ上で設置された検索エージェントが検索フォームを作成する．(ii) 検索ワードを Proxy エージェントに送る．(iii)Proxy エージェントが Web 検索エンジンに検索ワードを与え，検索結果を取得する．(iv) 検索エージェントは検索結果から，タイトル，URL を取得する．(v) 検索エージェントは Proxy エージェントから与えられたデータをユーザが閲覧中のページに表示する．実際の検索エージェントのシステム構成図は図 2 の通りである．

4 おわりに

本稿では，Web ブラウザ上で動作する JavaScript プログラムのマルチスレッド化，および応用としてマルチスレッド化を用いたマルチエージェントシステムの試作について述べた．JavaScript プログラムをマルチスレッド化することで次のような利点が挙げられる．第 1 に Web ブラウザ上で動作するマルチスレッド機能が必要な JavaScript アプリケーションが作成できる．第 2 に JavaScript プログラムのマルチスレッド化によるパフォーマンスのチューニングができる．また，JavaScript マルチエージェントシステムを利用することで，機能を分割したプログラミングが容易に可能となる．

参考文献

[1] Yujiro Fukagaya, Tadachika Ozono, Takayuki Ito and Toramatsu Shintani, ”MiSpider: A Continuous Agent on Web Pages ,” In the Proc. of the 14th World Wide Web Conference(WWW2005), pp.1008-1009, May. 2005.

[2] 伊藤正都，大囿忠親，新谷虎松，”JavaScript におけるマルチスレッド機能の実現とその応用”，FIT2005 第 4 回情報科学技術フォーラム講演論文集