Mobidget の移動・分散基盤の実現

5U－2

小山和也*、五味秀仁*、藤田悟*、山之内徹*、野田誠†
*NEC, †NEC ソフトウェア神戸
e-mail: {kazuya,gomi,satoru,yamanouchi}@ccm.cl.nec.co.jp,
noda@bsd1.kbnes.nec.co.jp

1　はじめに

Mobidget [1] はエージェント移動、エージェント間通信、分散透明オブジェクト操作を実現することにより、多様な分散システムを容易に記述する事を可能にするプログラム言語およびその実行環境である。

本稿では、これらの機能を実現する実行環境、Mobidget 仮想マシン (以下 VM) の実装について述べる。Mobidget ではエージェント移動、分散透明オブジェクト操作、エージェント間通信の個々の機能が利用できるだけでなく、それらをマルチスレッド環境等で同時に利用することが可能である。

以下では、まず MobidgetVM の概要ならびに特徴的機能の実現について述べる。次に、エージェント移動とリモートメソッドコールの同時利用を実現するためのメソッドコール実現機構について述べる。最後に本システムの実装ならびに考察について述べる。

2　MobidgetVM 概要

2.1　プロセス領域管理

MobidgetVM は、バイトコードにコンパイルされた Mobidget プログラムをロードし、実行する環境である (図 1)。

VM は、複数のプロセス領域を持ち、複数のプログラムを並行して独立に実行する。エージェント内に分散を持たないモデルでは、1 つの プロセス領域が 1 エージェントに相当する。プロセス領域はそれぞれ独立したクラス / オブジェクト空間、およびメソッドの実行状態であるスレッドを持つ。

個々のプロセス領域は複数のスレッドを持ち、複数の処理を並行して実行できる。全てのスレッドの実行状態は、スレッドフレームのスタックの形で、VM 上のデータとして管理される。

オブジェクトは、フィールド等を持つ通常のオブジェクトと、他のプロセス領域内のオブジェクトへの参照を表すリモート参照オブジェクトの 2 種類がある。スレッドは、基本的に同一プロセス領域内のオブジェクトしか操作する事は出来ないが、この参照オブジェクトを経由する事で、他の空間内のオブジェクトも操作する事が可能となる [2]。

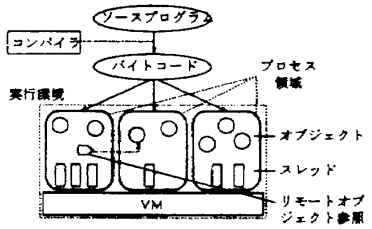

図 1: 概要

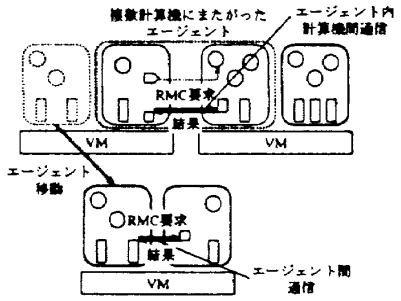

図 2: 特徴的機能

2.2　移動・分散・通信処理

MobidgetVM では、エージェント移動、分散透明オブジェクト操作、エージェント間通信などの機能が利用できる (図 2)。

エージェント移動は、プロセス領域データの転送として実現される。エージェント移動が発生すると、まず該当するプロセス領域上で動作する全てのスレッドを停止し、プロセス領域内上のオブジェクト、スレッド、クラス等のデータをシリアライズし、全て他の計算機にコピーする。同時に、元のプロセス領域は削除する。一方移動先ではコピーされたデータから元と等価なプロセス領域を再構築し、停止していた全てのスレッドの実行を再開する。

また、エージェントは、自身の一部のオブジェクトやスレッドを他の計算機に分散配置する事で、複数の計算機にまたがる事が出来る。この場合、それぞれの計算機毎に 1 つずつプロセス領域が作られ、エージェントはそれらの領域の集合として実現される。他のプロセス領域上のオブジェクトに対する操作は VM によって自動的にプロセス領域の間の通信 (リモートメソッドコール (RMC)) に置き換えられ、分散透明なオブジェクト操作環境が提供される。

エージェント間通信は、分散透明オブジェクト操作と同様の枠組みとして、他のエージェントを表すプロセス領域上のオブジェクト操作として実現される。ただし、

Implementation of mobile and distributed functions in Mobidget

Kazuya Koyama, Hidehito gomi, Satoru Fujita,Toru Yamanouchi, Makoto Noda
NEC Corporation, NEC Software Kobe,Ltd.

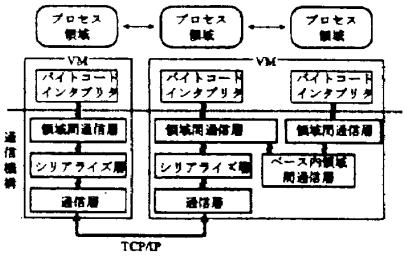

図 3: 通信機構

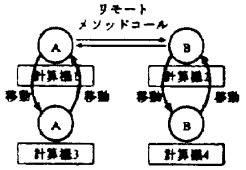

図 4: 移動エージェント同士の通信

行える操作はインタフェース経由のメソッドコールのみで、かつ領域間で渡せるオブジェクトは限定される。

VM は移動・分散機能専用の通信機構を内蔵しており、これを用いて領域間、VM 間の通信を行う (図 3)。通信機構は、バイトコードインタプリタへのインタフェースを提供する領域間通信層、スレッドやオブジェクトをシリアライズするシリアライズ層、他の VM への接続やデータの送受信を行う通信層からなる。また、同一 VM 上の領域間で通信を行う場合、ベース内領域間通信層を通して直接通信を行い、シリアライズ等の処理を省く事が出来る。

3 移動とリモートコールの同時利用

Mobidget では移動・分散処理をマルチスレッドで利用可能な事から、これらを組み合わせて、相互に移動しあうエージェントが互いにメソッドを呼び合うようなシステムも容易に構築する事が出来る (図 4)。これを実現するため、MobidgetVM は、プロセス領域内の実行状態だけでなく、プロセス領域間の通信状態もあわせて管理し、リモートメソッドコール実行中でのエージェント移動とその後の動作継続を可能にする。

以下では、リモートメソッドコールのコール先、コール元がそれぞれ移動した場合の動作についてのべる。

3.1 コール先の移動

リモートメソッドコールを受けているプロセス領域が移動する場合、通常のプロセス領域の移動と同様に、リモートメソッドコールのスレッドを含む全てのスレッドが移動先に転送される。

この時、リモートメソッドコールのスレッドは、そのコール元の情報をあわせて転送し、同時に、移動元はコール元にスレッドが移動した事を通知する。移動先ではそのまま移動した時点から実行を再開し、実行が終了したらコール元に実行結果を送信する。コール元はコール先 / 移動元から移動発生通知を受けた後、移動先から実行結果が送られて来るのを待ち、実行結果を受信した後は通常通り実行を継続する (図 5)。

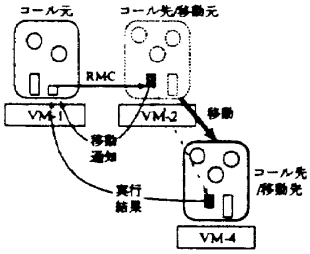

図 5: コール先の移動

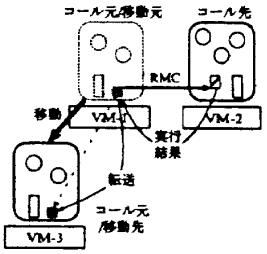

図 6: コール元の移動

3.2 コール元の移動

リモートメソッドコールを外部に行っているエージェント空間が移動する場合、通常のプロセス領域の移動と同様に、コール元のスレッドも移動先に転送する。ただし、移動元のスレッドは削除せず、エージェントの移動先を記録して、コール先から実行結果を受け取るまで待機する。そして、コール先が通常通りコール元に実行結果を送信すると、移動元はこれを移動先に転送し、自身を削除する。移動先のスレッドは、移動直後は実行結果受信待ちの状態で実行を再開し、移動元から実行結果の転送を受けたら、通常通り実行を継続する (図 6)。

4 おわりに

本稿では Mobidget の機能を実現する VM の実装について述べた。VM はプログラムの実行状態や通信の状態等を管理する事で、エージェント移動やリモートメソッドコール、およびそれらの同時利用を可能にした。

現在、MobidgetVM は試作版が完成しており、Windows95/NT 上で利用可能となっている。VM 自身は C 言語で記述し、スレッドは Win32 API のスレッドライブラリを用いている。

今後は、試作版の実行速度や通信方式の評価 / 改善を行うと共に、通信障害対策も改善していきたい。

参考文献

[1] 藤田, S.Jagannathan, R.Kelsey, 小山, J.Philbin, 山之内: “移動・分散プログラミング言語 Mobidget: 言語仕様”, 情処第 57 回全大, 1998.

[2] 五味, 小山, 藤田: “Mobidget の分散オブジェクト管理”, 情処第 57 回全大, 1998.