Java 言語における非同期通信の抽象化

阿部 修

北陸先端科学技術大学院大学 情報科学研究科

1997 年 2 月 14 日

キーワード: 計算機言語, 並行オブジェクト指向モデル, フューチャー, Java, 分散計算.

本研究の目的と背景

本研究では Java で並行計算をより抽象的に扱えるように言語を拡張し、そして、その言語を Java バーチャルマシン上で動作させることを目的とする。

現在、コンピュータネットワークが急速に普及し、並行計算の応用範囲も急速に増えてきている。しかし、これらの開発はほとんどの場合、未だに逐次計算のためのパラダイムに頼っている。とりわけ計算機言語ではそうである。例えば、アプリケーションの開発に実際に用いられる言語のほとんどが、逐次計算のために開発された言語にプロセスやスレッドまたはコルーチンといった機構を組み合わせることで実現している。

しかし、並行計算には、逐次計算にはない特有の問題が内包されている。例えば、同期のとり方やデータの共有の仕方、クリティカルセクションの排他制御などの問題がある。このような問題をすべて逐次計算のパラダイムだけに押し込むことは、最適な方法とはいえない。逐次計算のパラダイムで設計された言語を用いて、並行計算のためのプログラムを書くことは、人間にとってあまり直観的でなく、理解しにくいものとなってしまう。

Java というオブジェクト指向言語は Web ブラウザによってリモートマシンにあるプログラムをロードして実行することができる。 Java はそれだけで画期的な言語ではあるが、 Java もまた並行計算に適した言語ではない。また、RMI や HORB といった Java で分散オブジェクトを抽象的に扱うためのシステムは現れたが、並行計算を抽象的に扱えるようにするためのシステムは現れていない。分散計算においても並行計算の扱いやすさは重要なことである。

並行オブジェクト指向言語 MILK

本研究で作成した言語には MILK と名付けた。MILK は Java を拡張している。MILK では、並行計算を抽象的に扱うために、Java に並行オブジェクト指向モデルを組み込んでいる。並行オブジェクト指向モデルでは、各オブジェクトが自立して並行に動作することができる。従って、オブジェクトを擬人化して捉えることができる。このことから、人間にとって並行性を直観的に考えることがしやすく、プログラムの記述が容易になると考えられている。

並行オブジェクト指向モデルでは、オブジェクト間でメッセージによる非同期通信を行うことで、計算を進めていく。MILK ではメッセージ通信の方式に 2 種類の方式を採用している。それらの方式は過去型と未来型と呼ばれている。過去型はオブジェクト間でメッセージ通信を行った時に、送信側が受信側の処理の終了を待たずにすぐに計算を再開する方式である。過去型ではメッセージ送信後、受信側から戻り値が返されることはない。未来型はフューチャーと呼ばれるプレイスホルダを、送信側から受信側にメッセージと共に渡し、過去型と同様、送信側はすぐに計算を再開する。そして、受信側は処理の結果をフューチャーに渡す。送信側は結果が必要になった時にフューチャーを調べ、結果が戻っていればその値を使う。結果が戻っていない場合には、そこで戻ってくるまで待つという方式である。

MILK は本研究で作成したトランスレータで Java に変換することができる。トランスレータは、まずソースコードをスキャナでトークンの列に切り分ける。切り分けられたトークンの列はパーサに渡され、構文木が作成される。木の各ノードはオブジェクトで構成され、各構文の必要な情報を保持している。そして、トランスレータはこの構文木をたどり、Java から拡張した構文を発見すると Java で動作するコードに変換しながら、再び文字列にしていく。

変換は、並行オブジェクトの基本となる定義やフューチャーをあらかじめクラスライブラリとして用意しておき、MILK の Java の構文と異なる部分を、このクラスライブラリを利用して Java で動作するコードに置き換えていく。並行オブジェクトは皆、ライブラリで定義したこのクラスから派生させて定義する。このクラスではメッセージキューとスレッドを状態として持っている。並行オブジェクトはこの 2 つを使って非同期通信を行っている。また、フューチャーのクラス定義は正しく同期がとられるように設計してある。

分散環境では、そのシステムにあわせてこれらのクラスライブラリをあらかじめ作り変えておく。そして、変換時に使用するクラスライブラリを適切に切替えることで、そのシステムに適した変換を行う。こうすることで、どんなシステムでも分散オブジェクトをすぐに並行オブジェクトとして使うことができる。

本研究の評価と展望

MILK では並行オブジェクトと、過去型、未来型のメッセージ通信を Java に組み入れた。これにより、並行計算を人間の直観的な表現方法で実現できるようになった。また、

MILK は Java を拡張したため、 Java が持っている長所をそのまま受け継ぐことができた。もともと Java はプログラムの記述性容易性が高い言語であるが、MILK によりそれがさらに高まった。

しかし、本研究で実装したトランスレータでは名前の参照の解決は 1 つのクラス定義内だけで行うので、それに伴う制限もかなり多かった。例えば、他のクラスファイルで定義されているフューチャーの参照は解決できないので、それを禁止した。また、 Java 自体のオブジェクトの不備による並行オブジェクトの情報隠蔽の不完全さも問題である。

このような欠点は、トランスレータで Java ソースコードに変換する方式では無く、コンパイラを作成し、直接 Java クラスファイルを生成する方式にすることで、技術的には解決することができる。また、コンパイラで実現する方法は Java の制約にとらわれないので、より柔軟な構文で並行計算を表現することも可能である。

現在、 Java の VM はパラレルマシン上では実装されていない。従って、並行オブジェクトは本当の意味でその真価を発揮することはできない。しかし、近年のハードウェアの進歩と普及の速度を考えるとパラレルマシンが普及してくる日もそう遠くはないと考えられる。そうなってくると、MILK のような並行オブジェクト指向言語もその活躍の場を増やしていくことだろう。