Meta-leval Architecture を用いた移動計算機環境を支援する言語処理系の実現

天野 憲樹
北陸先端科学技術大学院大学
情報科学研究課 情報システム学専攻

1996 年 2 月 15 日

キーワード

移動計算機環境，メタレベル ・アーキテクチャ，
総称関数，メタオブジェクトプロトコル

論文要旨

本研究で我々は計算機の移動等に伴うシステム構成や環境の動的な変化に柔軟に対応可能なアプリケーションを作成 ・支援するための枠組として、メタレベル ・アーキテクチャを用いた言語処理系を提案し、実装した。

本研究の背景

近年の技術革新は電子装置の小型化、高性能化をもたらし、その製造技術の進歩はそれらの安価な実現を可能にした。その結果、パーソナルユースの携帯電話、ポケットベル等が急速に普及するこことなった。またネットワークは拡大の一途を辿り、オフィスはもとより、パソコン通信などを通じて急速に個人の生活空間に浸透しつつある。そして無線通信の技術革新により携帯可能な小型コンピュータの実用性が高まってきた。このような携帯可能な小型コンピュータとネットワーク、そして無線通信という 3 つの技術の融合により新しいコンピューティング環境、コンピューティングスタイルが生まれようとしている。それは今までのコンピュータの利用形態とは全く異なる利用が可能になり、今までにないコンピューティングスタイルが生み出される可能性を含んでいる。このように移動可能な小型計算機をネットワークに接続し、分散環境と融合させた計算機環境を移動計算機環境という。

このような移動計算機環境に内在する問題としてバッテリ容量の制約や物理的な位置に依存している現在のネットワークアドレス (IP アドレス)、そして無線通信における狭いバンド幅や高いエラーレート、そして一時的な回線切断等があげられる。またこうした環境ではその時々の計算機の使用形態、使用場所、使用目的に応じて、計算機システムの構成や環境が動的に変化する。しかし計算機システムの動的変化に応じてその上で動作するアプリケーションまでが動的に変化するわけではない。これは現在のコンピュータ及びネットワークが移動計算機環境のような環境を想定して実現されていないことに起因する。つまり計算機システムの構成や環境が動的に変化

[0]Copyright ©1996 by Noriki AMANO

する可能性があるのに対して、アプリケーションは静的である。これが移動計算機環境におけるシステムの透過性やアプリケーションの効率的実行を困難にしている要因であると言える。移動計算機環境におけるシステム構成の動的変化に対応する枠組として従来の研究は、動的な変化により生じるシステム上の差異を OS もしくはそれに近いレベルで吸収することでアプリケーションに対するシステム透過性の実現やアプリケーションの実行効率の維持を目指すものが多い。しかしこれだけで十分であるとは言えない。なぜならば OS による汎用的なサービスで個々のアプリケーションの振舞いをサポートすることには限界があり、かつ完全なシステム透過性の実現は現実的でないからである。このような環境においてはシステム構成や環境の動的変化に応じて振舞いを変えるアプリケーションが望まれる。

本研究のアプローチ

移動計算機環境ではホストの移動等によるシステム構成の動的変化に柔軟に対応することが困難である。こうした環境にはリフレクションやメタレベル・アーキテクチャといった技術の応用が可能であり、本研究では、このようなシステム構成の動的な変化に柔軟に対応可能なアプリケーションを作成・支援するためにメタレベル・アーキテクチャを用いた言語処理系を提案する。本研究のアプローチは OS レベルによるシステム、あるいはネットワークの透過性を実現することではなく、アプリケーションの振舞いをシステムの状況に応じて積極的に変化させることによりシステム構成の動的な変化に対応しようというものである。つまりシステム構成の変化により生じたシステム上の差異をアプリケーションの振舞いによって吸収する。しかしシステムの透過性が必要でないということではない。OS やそれに近いレベルでシステムの透過性を実現しつつ、かつそれに応じてアプリケーション自身もその振舞いを変化させることでこのような環境に対応するということである。本研究はこのアプリケーション自身の振舞いに焦点を当てたものである。

本研究で提案する言語処理系はアプリケーションに対して空の枠組を提供する一種のサーバと位置付けられる。各アプリケーションの状況に応じた振舞いはこの空の枠組の中に記述される。そしてこの言語処理系が現在のシステム構成に応じた振舞いを各アプリケーションに割り振るサーバとして機能する。この機構により環境に適した個々のアプリケーションの振舞いをサポートすることが可能になる。

言語 LEAD

このようなシステム構成の変化に応じたアプリケーションの振舞を記述するためにメタレベル・アーキテクチャを用いたプログラミング言語 **LEAD** を提案する。アプリケーションをベースレベルとし、その中で計算機システムの構成や環境に依存する処理 (メソッド) を総称関数としてメタレベルに言語 LEAD で記述し分け、言語 LEAD の実行系 (インタプリタ) が、システムの構成や環境に即してメソッドのディスパッチを行なう。これによりアプリケーションプログラムのうち、システム構成及び環境に依存する処理部分を状況に応じて選択し、実行することが可能になる。このようなメソッドディスパッチの制御部分は一種のメタオブジェクトプロトコルとして提供されている。つまりこれは言語 LEAD によって記述され、LEAD の関数としてユーザーに公開されており、ユーザーのカスタマイズを可能とする。メソッドディスパッチのポリシーを LEAD によって動的に記述可能であり、その内容は動的に LEAD インタプリタに反映する。また LEAD はプログラムのリモート実行を行なう機能を内蔵し、負荷のかかる処理を高い処理能力を持つリモートホスト上で実行することにより、システム構成・環境によらないアプリケーションの実行効率を維持する働きも持つ。このようなリモート実行は LEAD インタプリタにより状況に応じてな

される。また LEAD インタプリタは言語処理系でありかつシステム構成 ・環境の動的変化に対応する処理を制御するサーバのように見える。socket ベースのプロセス間通信機能を持つインタプリタ言語で記述されたアプリケーションであれば LEAD インタプリタと通信することによって、システム構成 ・環境に依存した振舞いを実現する機能を付加することができるからである。

結論

システム構成 ・環境の動的変化が著しい計算機環境に対し、本研究で提案 ・実装した言語処理系 LEAD の有効性を以下にあげる。

- システム構成 ・環境の動的変化に対応する部分をアプリケーション本体から分離して記述できる。 (モジュール化)
- システム構成 ・環境の動的変化に応じたアプリケーションの振舞いを選択する共通の機構をアプリケーションに対して提供する。
- 既存の言語からなるアプリケーションに、その言語処理系やアプリケーション自体を変更することなく、システム構成や環境に依存した振舞いを実現する機能を付加することができる。
- メソッドの実行ソースコード部分の記述は特定の言語に限定されない。
- メタオブジェクトプロトコルとしてメソッド選択のポリシーをユーザーに公開しているので、システム構成 ・環境に応じてこのポリシーを柔軟にカスタマイズすることができる。
- システム構成 ・環境の状態を示す関数は LEAD では通常の関数と同様に呼び出すことができる。つまり LEAD インタプリタ上にダミーの関数を作成することで、あるシステムの状態 (構成 ・環境) を容易かつ動的にシミュレートすることが可能となる。
- プログラムのリモート実行を行なうことでシステム構成 ・環境によらずアプリケーションの実行効率を維持することができる。