4U-06

ネットワーク結合による統合実行環境の実現

高橋 右[1,a)]　林原 尚浩[1,b)]

概要：プログラムの開発環境を構築する際に，プログラマの負担が大きいことが多い．パッケージ管理システムでは簡易的に構築することが可能であるが，グループ内（会社や教育機関など）で環境を共有する際に，インストールに手間と時間がかかり，すぐに導入することができない．一方，PaaS ではこの問題点を克服するため，提供する開発環境をインストールなしで使用することができる反面，自社製フレームワークを PaaS として利用できないという点，アーキテクチャ依存による命令が利用できない点，多くの PaaS ではサポートされる開発環境が比較的少ないという点などの問題点から，社内で PaaS サーバを構築する場合が多い．そこで，本研究ではネットワーク上に構築した統合実行環境において，P2P ネットワークによってユーザ同士が開発環境を共有し合い，PaaS サーバも必要としない統合実行環境を実現する．

キーワード：分散実行環境，PaaS，Java，RMI

1. はじめに

1.1 背景と問題点

プログラムの開発環境を構築する上で，パッケージ管理システムや PaaS によって，簡易的に開発環境（ライブラリやフレームワーク）を利用できるようになってきている．パッケージ管理システムでは手元に開発環境をインストールして環境構築できるが，グループ内（会社や教育機関など）で環境を共有する際に，インストールに手間と時間がかかり，すぐに導入することができない．一方，PaaS ではインストールなしで開発環境を利用できるが，自社製フレームワークを PaaS として利用できないという点，アーキテクチャ依存による命令が利用できない点，多くの PaaS ではサポートされる開発環境が比較的少ないという点などの問題点から，社内でコストのかかる PaaS サーバを構築しなくてはならない場合が多い．

1.2 研究の目的

本研究では，グループ内（会社や教育機関など）で環境を共有する際に，メンバが新たに加わった場合でも，ライブラリ管理を意識させずに，すぐに必要とする開発環境を導入することができ，またサーバのコストを必要としないシステムを実現することを目的とする．

[1] 京都産業大学
603–8555，京都市北区上賀茂本山
[a)] lightfox.task@gmail.com
[b)] naohaya@cse.kyoto-su.ac.jp

2. 関連研究

2.1 複雑なグリッド環境で柔軟なプログラミングを実現するフレームワーク

通常のグリッド環境用フレームワークでは，プログラミングの柔軟性を犠牲にしてしまい，プログラマに大きな負担をかけてしまう．弘中ら [1] は，グリッド環境でのプログラミングは複雑であることから，これらの問題点を解決できるフレームワークを提案した．

2.2 クラスタ設定のパッケージ化の設計と実装

多数のマシンの OS やソフトウェアの設定を自動的に行う自動インストーラを利用する際に，小規模なクラスタシステムでも 1 ノードあたり数百〜数千行もの設定ファイルの作成する必要があるなどの理由から，利用されていないとされている．そこで，高宮ら [2] は複数のパッケージをグループに分類させることで，自動インストーラの設定作業を大幅に簡略化する仕組みを実装している．

3. 提案システム

3.1 概要

本研究では，Java 言語のライブラリを対象とする．提案システムでは，P2P ネットワークで接続された端末において，参加した端末がライブラリの管理・共有を意識せずに，容易にライブラリを導入することができる．

3.2 ライブラリの管理と共有

図1に提案システムの全体図を示す．図1の各ノードに

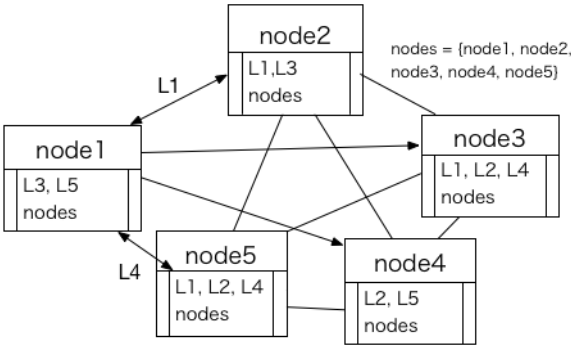

図 1 ネットワーク結合による統合実行環境の全体図

はインストール済みのライブラリ（L1〜L5），ノードのリストを表すnodesの情報を持つ．提案システムは，以下に示すライブラリのリモート実行，登録，インストールの3つの機能から成り立つ．

3.2.1 ライブラリのリモート実行

ライブラリのリモート実行をする際には，ユーザプログラムがライブラリを遠隔で使えるようにするために，JavaRMI[3] を使用する．JavaRMI とは JavaVM の間で RPC(Remote Procedure Call) によって，サーバからクライアントに対してオブジェクトを遠隔で転送する機構であるが，呼び出すオブジェクトを各ライブラリのオブジェクトとすることで，ユーザプログラムが別のマシンにある開発環境を利用できるようにする．そのため，ユーザプログラムがリモート実行するには，JavaRMI を使った通信用プログラムに変換する必要がある．この通信用の変換は変換器 rlibc（rlibc コマンド）によって行う．

変換器 rlibc の実装方法として，ユーザのソースコードをパーサジェネレータ ANTLR[4] を用いて構文解析し，ソースコードを通信用プログラムに変換する．ANTLR とは Lex，Yacc に比べて，構文木を自動生成できるパーサジェネレータである．また，rlibc によって変換を行う際は，図1にあるように各ノード（端末）に対して，必要なライブラリが存在するかを全ノードにブロードキャストし，その結果から，必要なライブラリのあるノードのホスト名，ポート番号を変換に反映させ，コンパイルしリモート実行を行う．ただし，全ノードにブロードキャストしているのは簡易化のためであり，今回は全ノードが密に接続されていることを前提とする．

3.2.2 ライブラリの登録

ライブラリの登録は，今回は Java 言語を対象としているため，必要なライブラリのパス（jar ファイルのパス）を指定し，登録を行う．登録に関してもライブラリを使用する際と同様に，rlibc コマンドを使用し登録を行う．jar ファイルからライブラリのメソッド名，引数の型などの情報を取得し，その情報を元に自動的にサーバを立ち上げ，ライブラリを使用できるようにする．

3.2.3 ライブラリのインストール

本研究の統合実行環境は PaaS の拡張でありリモート実行を主要とするが，ユーザは開発環境をローカル環境で実行する必要がある場合も多く，ローカル端末に実行環境を構築する必要もある．また，多くの PaaS ではインストールするサポートは少ないため，本研究ではインストール管理も行う．インストール方法はリモート実行と同様に rlibc コマンドを使用するため，ユーザプログラムから自動的に必要なライブラリを探し，インストールすることが可能である．また，リモート実行とも連携しており，ユーザがライブラリを何度もリモート実行する場合は自動でインストールされる．

3.3 利点

PaaS のようにサーバ側で編集するのではなく，JavaRMI によって複数の端末からライブラリを補完し合い，実行出来るため，多くのライブラリをサポートでき，アーキテクチャ依存の命令も利用できる．また，オリジナルのライブラリであってもそのライブラリを登録し利用できるため，ローカルのネットワークで共有したい場合でも利用できる．そして，ユーザのプログラムから自動的に必要なライブラリを探すため，パッケージマネージャよりも簡易にインストールが可能である．

4. まとめと考察

現時点では，ユーザが実行環境を取得する際に，近隣ノードに対してブロードキャストし，問い合わせを行っているが，今後は実行環境に必要なメモリ，処理性能に適したルーティングを実装を考えている．また，ライブラリ共有だけでなく，フレームワークやコマンドのような開発環境も視野に進めようと考えている．

参考文献

[1] 弘中 健，斎藤 秀雄，高橋 慧，田浦 健次朗：複雑なグリッド環境で柔軟なプログラミングを実現するフレームワーク，情報処理学会論文誌，p157-168 (2008).

[2] 高宮 安仁，松岡 聡：クラスタ設定のパッケージ化の設計と実装，情報処理学会研究報告. [ハイパフォーマンスコンピューティング] ，pp.55-60 (2004).

[3] ORACLE, Remote Method Invocation Home(online), available from ⟨http://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html⟩ (accessed 2014-11-14).

[4] PARR, T., ANTLR(online), available from ⟨http://www.antlr.org⟩ (accessed 2014-11-14).