大規模分散アプリケーションの実証を可能とする PIAX Testbed の実装

寺西 裕一[1,2,a)]　下條 真司[2,1]

概要：我々は，大量のノード（コンピュータ，サーバ）実機上でのを用いた並列分散処理を行なうアプリケーションの実証実験を簡単化し，少ない手間で検証や評価を行なうことを可能とする PIAX Testbed の設計と開発を行なっている．PIAX Testbed では，利用者が専用のパッケージを作成すれば，一台の計算機上のエミュレーション，大量数のネットワーク上の分散ノード群それぞれにおいて，実行環境の違いを意識することなく，短期間で開発・デバッグ・分散実行できる．また，実験中，複数の遠隔ノード上で実行中のプログラムに対する操作を，新規の操作インタフェース等の開発をすることなく，インタラクティブに行なえる．本稿で示した PIAX Testbed は JGN-X 上で実装・運用を行なっている．

1. はじめに

近年，クラウド技術を中心として，データの保存や処理を 1 つのノード（コンピュータ，サーバ）に集中させず，並列分散処理を行なう分散アプリケーションの構成方法に関する様々な技術が提案されている．

一般に，分散アプリケーションが並列分散処理を行なう目的として以下があげられる．

- 1 つのノードで実行した場合に長い処理時間が必要となる複雑な処理を分割し，並列実行することで処理時間を短縮させる
- マルチユーザ環境でユーザ毎の処理を複数ノード上で並列実行することでノードあたりの負荷を減少させ，同時に収容可能な利用者数を増加させる

したがって，並列分散計算処理を行なうためのアルゴリズムやアプリケーションの構成法に関する研究の有用性は，処理時間が短縮可能であるかどうかや，処理負荷をどの程度低下させられるかによって明らかにされるべきものであり，複数の実機ノードを用いた実験が不可欠となる．データセンタ間クラウドシステムへの適用性を実証するには複数のデータセンタに跨った大規模な実験が必要となる．

しかし，そのような大規模環境を一般の研究者が用意することは容易ではない．したがって，従来，この研究分野では，主にシミュレーションによって動作や性能が確認されてきた．しかし，アルゴリズムやアプリケーションの実機ノードにおける処理時間や処理負荷は，シミュレーションのみでは確認できない．実際のノード上で動作するソフトウェアでなければメモリ使用量，CPU 負荷等の実用性に関わる評価もできない．また，実機ノード間の通信遅延の違いによってデッドロックが生じる等の問題がアルゴリズムにあったとしても，シミュレーションでは再現できない場合がある．

このため，これまでに研究者間で計算機資源を持ち寄り，仮想化して共有することで，多くの研究者が大規模な実験を行なう環境 [1][2] の研究開発がなされてきた．しかしながら，分散アプリケーションを実機上で動作させるには，ノード毎・アプリケーション毎の実行環境を構築し，ノード間の通信設定を環境毎に都度行なうなどしなければならず，また，デバッグは，都度アップロードしたプログラムを遠隔で分散実行しながら行なねばならないなど，実験者の手間は大きかった．

我々は，大量のノード上での分散アプリケーションの実証実験を簡単化し，少ない手間で評価・検証することを可能とするテストベッド「PIAX Testbed」[3] を開発し，JGN-X[4] 上の計算機資源を用いた運用を 2013 年より開始した．PIAX Testbed は，自律的にアプリケーションレベルのネットワーク (オーバレイネットワーク) を構成する Java ベースのミドルウェア PIAX[5] を基盤とする．PIAX によって，実験者は複数ノードを用いた様々な種類の分散アルゴリズムやアプリケーションを，ノード間のネットワーク構成や実装の詳細を知ることなく実装することができる．

我々はこの PIAX を用いた分散アプリケーションを実行・操作するテストベッドの利用モデル，ならびに，環境独立なアプリケーションパッケージ設計とその実行環境を提案する．提案する利用モデルは，環境の構築や設定を実験者から分離し，実験者は実行環境を意識することなく，

[1] 独立行政法人情報通信研究機構
National Institute of Information and Communications Technology, Japan
[2] 大阪大学サイバーメディアセンター
Cybermedia Center, Osaka Uniersity, Japan
[a)] teranisi@nict.go.jp

分散アプリケーションの開発と実行に注力することができる．また，提案するパッケージ設計および実行環境によって，PIAX Testbed は，デバッグ，実行比較，プログラムの遠隔操作といった分散アプリケーションの研究開発過程での作業効率の向上が可能となった．

2. 並列分散処理の実証環境の要件

本章では，大量のノードを用いた分散アプリケーションの実験環境において要求される要件，および，UNIX システムとして構成された複数のノード上でプログラムを実行する一般的な実験環境において生じる課題を列挙する．

(1) 実行準備・起動
分散アプリケーションの実験を実機ノード上で行なうには，環境の整備，プログラムの配備，遠隔環境における起動といった作業を複数ノード上で実行する必要がある．実行準備や起動に関わる設定や手間は，実験者にとって本質的に必要な作業ではなく，なるべく削減できる必要がある．一般的な UNIX システムにおいては，実験者が各プログラムをノードへ FTP や SCP 等の手段で送信し，実行することが基本である．プログラムをいずれのノード (IP アドレス・ポート番号等) で実行するかは，あらかじめ設計のうえ，ノード毎に起動設定を行なう等の作業が個別に必要となる．こうした設定は，通常アプリケーション毎に個別に行なう必要があり，実験者の手間が大きい．知識の少ない実験者は，この段階をクリアすることすら難しい．

(2) デバッグ
分散アプリケーションのプログラムに限らず，プログラム開発にはデバッグが不可欠である．しかし，分散アプリケーションの場合，特に，プログラムが遠隔環境で，かつ，ノード間で連携動作するため難易度は高い．大量のノードを用いる実験環境では，分散動作するプログラムの動作を追うことは難しいうえ，不具合の原因を取り除き，再度プログラムを FTP や SCP 等の手段で送信し直す作業を，実験者が繰り返さねばならない．

(3) 実験中の動作変更
分散アプリケーションの実験においては，例えば，動作を開始させる，動作を変える，動作を停止させる等の操作を，ノードの部分集合に対して操作を試行錯誤に応じて実行する必要性がしばしば生じる．遠隔で動作中のプログラムの動作を変更したい場合，例えば，UNIX システムにおいては，遠隔・外部から SSH 等の手段でホスト名や IP アドレスを指定してログインし，コマンド実行により操作を行なう等の方法を取る必要がある．大量のノードを用いる場合，このような操作の手間は大きい．

実ノードを用いた大規模な実証を行なえる既存の実験環境の一つとして，PlanetLab[1] がある．PlanetLab は，仮想化技術によって，参加者がコンピュータリソースを持ち寄って，参加者同士で共有する仕組みを提供する．実験者は，仮想化された自分専用の OS が利用可能である．

PlanetLab では，大量のノードを用いたプログラムを動作させる必要があり，上記「(1) 実行準備と起動」をサポートする様々なツールが公開されている．PlanetLab Experiment Manager[6] は，プログラムの配備，実行やモニタリングを行なうためのツールである．こうしたツールは，プログラムの配布の手間を軽減するが，「(2) デバッグ」,「(3) 遠隔操作」の課題を十分に解決するものではない．

「(2) デバッグ」について，文献 [10] は，GUI を介した操作を一括して行なう GUI のエージェントを各ノードに具備させる設計により，一括した操作とデバッグを行なえる仕組みを提案している．しかし，この方法は，デバッグ自体を遠隔にプログラムを配備した状態で行なう必要があり，プログラムにミスがあった場合，再度プログラムを修正・コンパイルしたのち，配布・起動し直す必要がある．また，一般的に行なわれている IDE 等を用いたインタラクティブなデバッグは不可能である．

「(3) 遠隔操作」について，SSH を通じた実行の労力や作業量を軽減するためのツールとして，pShell[7], pssh[8] 等がある．これらは，複数のノード上でのコマンド実行を一括して実行することができる．しかし，実行中のプログラムを操作するには，コマンドラインから操作可能とするためのインタフェース開発が別途必要となる．また，ノードの一部集合に対して操作を行ないたい場合，各ノードがどのような役割を果たすかを覚えておき，IP アドレスやホスト名を直接指定して実行ノードを設定する必要がある．

3. PIAX Testbed の設計

PIAX Testbed は，以下の特徴により，前節に挙げた各課題の解決をはかっている．

(1) プログラム実行環境の分離
実験者が，PIAX[5] をベースとするプログラムを含むパッケージの作成・登録・実行のみを行なうモデルを採用する．実験プログラムには PIAX および PIAX が前提とする Java を用いるという制約が生じるが，PIAX Testbed 専用の SDK を用いてプログラムを記述し，後述の PIAX Testbed のパッケージ設計に基づいたパッケージを作成すれば，実験者は，プログラムの実行に関わる設定を意識することなく，実験環境を利用できる．

(2) 複数ノードエミュレーション環境の提供
1 台のローカル環境のコンピュータ上で複数ノードの実行のエミュレーション実行を行なえる環境を提供する．また，遠隔の実験環境での実行を，プログラムを一切変更することなく可能なパッケージ構成とする．これにより，ローカル環境で IDE 等を用いてプログラムのデバッグを行なったのち，同じプログラムを再コンパイル等することなく遠隔の実験環境で動作させることが可能となる．

(3) 発見型メソッドのインタラクティブな操作
実験者が，任意のノード集合に対し，実行中のプログラムが持つ Java クラスのメソッドを遠隔から実行する機能を提供する．この機能自体は PIAX が持つ機能

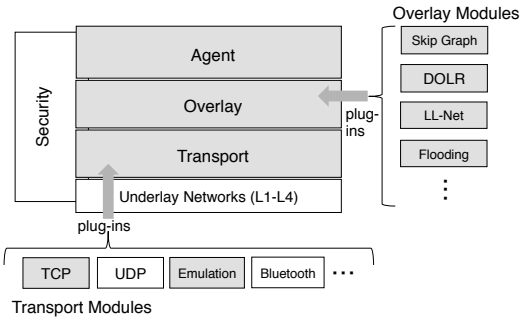


図 1 PIAX のコンポーネント構成

であり，アプリケーション開発に用いることができる API であるが，これを実験者が任意のタイミングでインタラクティブに利用・実行可能とする．これにより特別な遠隔操作用インタフェースを別途開発することなく，また，分散配置されている位置 (ホスト名，アドレス等) を意識せず，属性指定により動作中のプログラムを遠隔操作可能である．

上記それぞれの特徴について説明する前に，まず，本テストベッドが基盤として用いているミドルウェア PIAX について，その概要を説明する．

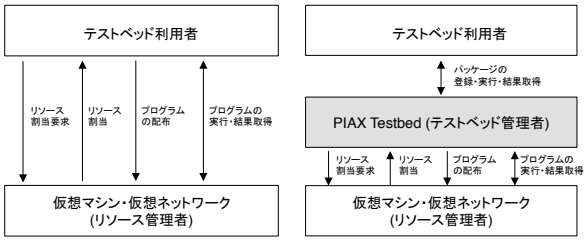
3.1 PIAX のコンポーネント構成

PIAX はオーバレイネットワーク上に「エージェント」と呼ばれるソフトウェアモジュールを動作させ，分散アプリケーションを実現するミドルウェアである．PIAX においては，ノード上のプロセス (スレッド) をピアと呼ぶ．すなわち，一つのノード上に複数のピアが動作する．**図 1** は，PIAX のコンポーネント構成を示している．PIAX は，大きく分けると Transport, Overlay, Agent という 3 つの機能要素から成る[*1]．

Transport は，下位ネットワーク機能を抽象化する．具体的には，ノード間の通信を，通信端点間で send および receive を実行するコンポーネントとして抽象化する．実装としては，TCP, UDP, Emulation, Bluetooth 等がある．PIAX Testbed では，実ネットワークにおけるノード間の通信に TCP/IP のコンポーネントを用いる．通信端点としては，IP アドレスとポート番号の組が用いられる．また，PIAX は単一 PC 上で複数のネットワークエミュレーションのコンポーネント実装 (Emulation) を含んでいる．

Overlay は，Transport のコンポーネントを用いて実現されるオーバレイネットワーク機能である．PIAX は，オーバレイネットワークアルゴリズムをプラグインとして組み込むことができる．構造化オーバレイネットワーク Skip Graph [11] および，いくつかの派生アルゴリズム実装 [12], [13] や，フラッディングを行なう実装があらかじめ組み込まれており，ピアが持つ属性情報 (キー) の完全一致検索，範囲検索，2 次元範囲検索，フラッディング探索などが実行可能である．

[*1] Transport, Overlay, Agent 以外に，Security の機能要素があるが，PIAX Testbed ではとくに利用しない．



(a) 既存の分散アプリケーション実証環境の基本構成　(b) PIAX testbed

図 2 PIAX Testbed の利用モデル

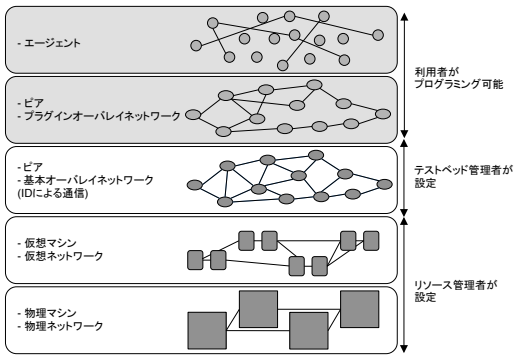


図 3 PIAX Testbed の機能レイヤ

Agent は，ピア上で動作するソフトウェアモジュールである．ピア上に複数のエージェントを動作させることができる．各エージェントは属性情報として，名前・位置などを保持する．PIAX では，オーバレイネットワークが持つ検索機能を用いて，データ配信やデータ集約の対象となるエージェント集合を，この属性情報の条件によって指定できる．どの属性を，どのオーバレイネットワークを用いて検索可能とするかは，属性毎にプログラム中で設定する．

エージェント間でのデータ配信やデータ集約は，遠隔メソッド呼び出しによって行なう．PIAX は，相手のエージェントの識別子を直接指定して通信する「遠隔メソッド呼び出し (PIAX においては call と呼ばれる)」の他，オーバレイネットワークを用いて検索されるエージェント集合に対するメソッド呼び出しを一括して実行する「発見型メソッド呼び出し (PIAX においては discoveryCall と呼ばれる)」の機能を提供している．PIAX を用いる分散アプリケーションは，基本的にこれらの遠隔メソッド呼び出しの機能を用いて並列分散処理を行なう動作を実装する．

3.2 PIAX Testbed の利用モデル

従来の実行環境における設定の手間を軽減するため，PIAX Testbed では，管理者によってあらかじめ設定がなされたピア上で，実験者者が PIAX SDK を用いて作成したパッケージを実行する利用モデルを採用する．これによって，実験環境の設定を専門の管理者の作業として切り出せるようにする．PIAX Testbed は，PlanetLab 等と同様に，物理マシン・物理ネットワーク上で仮想マシン・仮想ネットワークが動作する環境を想定する．仮想マシン・仮

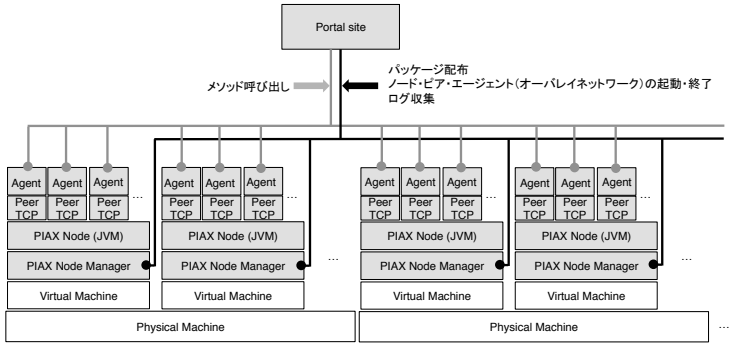


図 4 PIAX Testbed の全体構成

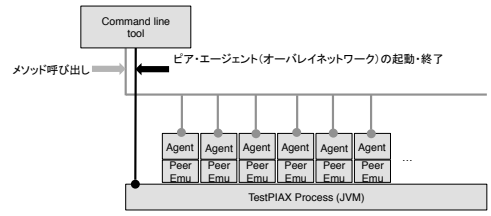


図 5 TestPIAX エミュレーションツール

想ネットワークは，リソース管理者システムおよびリソース管理者によって管理され，ユーザ毎にリソースが割り当てられる．

図 2 は，従来の分散アプリケーション実行環境，および，PIAX Testbed の利用モデルの違いを示している．図の通り，PIAX Testbed では，実験者となるテストベッド利用者とリソース管理者との間に，テストベッドシステムおよびテストベッド管理者が介在する．

PIAX Testbed の利用者は，実験用のエージェントを開発し，「パッケージ」として登録することで，実環境上で自分のプログラムを用いた実験を行なう．利用者に対するリソースの割り当ては，テストベッド管理者が行なう．基本的に，利用者毎に割り当てられた仮想ネットワークおよび仮想マシン上で，PIAX のピアが起動可能な状態にセットアップされた環境が利用者に解放される．利用者は，プログラムのパッケージをテストベッドに登録し，起動・利用・終了するのみであり，実行環境について意識する必要はない．

3.3 PIAX Testbed の構成

図 3 は，PIAX Testbed の実行環境における機能レイヤを示している．PIAX Testbed 利用者が，分散アプリケーションのプログラムとして記述するのは，上位の 2 つのレイヤ，すなわち，エージェントおよびプラグインオーバレイネットワークのレイヤである．各利用者向けには，ID によりピアの検索を行なえる基本オーバレイネットワークが構成され，このレイヤまでが，テストベッド実行環境として提供される．

PIAX Testbed においては，仮想マシン・仮想ネットワークのレイヤ以下は，利用者から直接操作されることは基本的にない．仮想マシン・仮想ネットワークは，実行環境としてテストベッド管理者が設定する．PIAX Testbed を PlanetLab 上で動作させる場合，このレイヤが PlanetLab に相当することになる．

図 4 は，PIAX Testbed のノード構成である．各仮想マシン上に PIAX Node と呼ばれるプロセス (Java VM プロセス) の起動や終了を行なうための PIAX Node Manager と呼ばれる常駐プロセスが動作する．ポータルサイトから

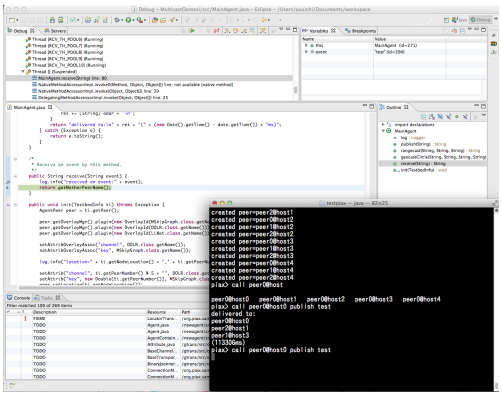


図 6 TestPIAX と Eclipse によるデバッグの例

のパッケージの配布，ピアの開始，終了の指示，ならびに，ログ等の収集は，この PIAX Node Manager を経由して行なわれる．

PIAX Node プロセス上には，複数のピアが動作する．1 つのピアは 1 つの Java VM 上のスレッドとして動作する．利用者が作成したパッケージは，それぞれの PIAX Node 上に配布されたのち，ピア上のエージェントとして起動され，ID による基本オーバレイネットワーク，および，設定されたエージェントの属性に基づくオーバレイネットワークが構成される．

3.4 TestPIAX: テストベッドエミュレーションツール

我々は PIAX Testbed における複数ノード上でのパッケージ動作のエミュレーションを 1 台のコンピュータ上で実行する TestPIAX と呼ばれるツールを開発した．**図 5** は，TestPIAX の構成を示している．TestPIAX は，PIAX の Emulation の Transport モジュールを用いてネットワーク機能をエミュレーションしており，1 台で複数のピアを起動させられる．メソッドの呼び出しや出力は，コマンドライン上で行なう．複数のノードをローカル環境で構築するため，プログラムを PIAX Testbed に登録し直すことなく修正実行できる．また，JDWP (Java Debug Wire Protocol) プロトコルにより，Eclipse 等の IDE を用いたプログラムのインタラクティブデバッグも可能である (**図 6**)．

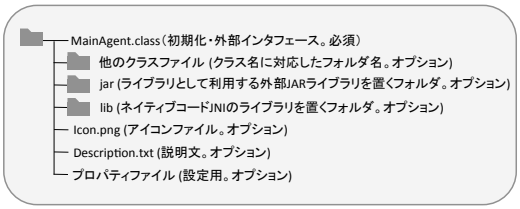


図 7 PIAX Testbed のパッケージ構成

Windows 環境では，パッケージファイルを，TestPIAX プログラムへドラッグ&ドロップすることで，エミュレーション実行することができる．

1 台の計算機で，大量の計算機上で動作するプログラムを実行することが可能なエミュレーションツールを提供する既存ソフトウェアとして，Overlay Weaver[9] がある．OverlayWeaver は，構造化オーバレイネットワークのアルゴリズムが用いる共通インタフェースを規定しており，同一のアルゴリズムコードを計算機 1 台上，および，実ネットワーク上で動作させられるが，一般的な並列分散処理を扱うものではない．

3.5 PIAX エージェントパッケージ

PIAX Testbed では，分散アプリケーションの実行に必要な情報を，単一ファイルにパッケージ化させた．ファイルフォーマットとしては，Java において一般的に用いられる JAR 形式を用いた．

図 7 は，PIAX Testbed におけるパッケージの構成を示している．各パッケージには，MainAgent と呼ばれるエントリポイントに相当するエージェントの実装を必須とする．パッケージファイルを異なる環境で共通で利用できるようにするためには，環境に応じた初期化設定を行なえる必要がある．そのため，MainAgent には，テストベッド専用の初期化関数を持たせた．MainAgent は，TestbedMainAgentIf と呼ばれるインタフェースを実装し，以下のインタフェースを持つ．

```
void init(TestbedInfo info)
```

メソッド名	動作
getPeer	ピアのインスタンスを取得する
getNodeName	ノード名 (ホスト名) を取得する
getNumberOfPeers	全体で起動されるピア数を取得する
getPeerNumber	ピアの起動順 (何番目か) を取得する
getNodeProperty	ノードが持つ属性を取得する
getPeerProperty	ピアが持つ属性を取得する
getNodeLocation	ノードの物理的位置座標を取得する

表 1 TestbedInfo のインタフェース

このインタフェースを通じてテストベッド上の初期化設定を行なうが，TestbedInfo は，表 1 に示すメソッドを持ち，MainAgent が起動されるピアや PIAX Node がテストベッド上で持つ属性を得ることができる．これらを用いて MainAgent を初期化することで，実行環境によらない，または，実行環境に応じた初期化定義が可能となる．これによって，エミュレーション環境と実環境，割り当てノード数が異なる環境において，共通のパッケージを定義できる．

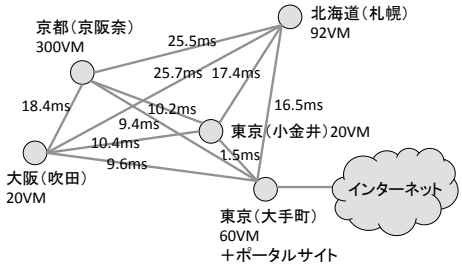


図 8 JGN-X 上への PIAX Testbed の配備状況および各配備拠点間の RTT

3.6 発見型メソッドのインタラクティブな操作

先に述べた通り，PIAX のエージェントは，「遠隔メソッド呼び出し」「発見型遠隔メソッド呼び出し」の機能を持つ．PIAX Testbed では，これら遠隔メソッド呼び出し機能を，利用者から，分散実行中のエージェントを遠隔操作するためのインタフェースとしても利用可能とした．

これにより，新たにコマンドラインインタフェースのような実行プログラムを開発することなく，Java クラスが持つ通常のメソッド定義を遠隔からテストベッド利用者が任意のタイミングで実行できる．また，発見型メソッド呼び出しによって，全体の一部のピアのメソッドを，条件指定により一括して実行するといったことが可能となる．

4. PIAX Testbed の実装と運用

我々は，情報通信研究機構 (NICT) が提供する高速広域ネットワークテストベッド JGN-X [4] 上のプロジェクトの一つとして，PIAX Testbed による実行環境を JGN-X 上の計算機，ネットワークリソースを用いて構築し，運用している．

4.1 運用の実際

図 8 は，PIAX Testbed で利用可能な仮想マシンが配備された拠点とその仮想マシン数，ならびに，各拠点間の RTT を示している．現在，札幌，京都，大阪，東京大手町，東京小金井にあるノード上で動作する仮想マシンを利用可能である (本稿執筆時点)．東京大手町にある NICT の計算機設備内に PIAX Testbed のポータルサイトが稼働しており，利用者は，インターネット経由でアクセスし，パッケージの登録や実行を行なえる．

利用者が PIAX Testbed ポータルサイトを経由してアカウントを作成すると，テストベッド管理者にメールで連絡される．テストベッド管理者がサイト上で承認すれば，アカウントが作成される．この間に，必要に応じて，NICT と実験者の間で共同研究契約を結ぶ手続きのやりとりを行なう．アカウント作成ののち，テストベッド管理者は，JGN-X 上のリソース (仮想マシン) を利用者ノードとして割当てる．

(a) ログイン画面　(b) ピア割り当て画面

(c) パッケージ操作画面　(d) エージェント操作画面

(e) 呼び出しグラフ　(f) 実行コンソール

図 9　PIAX Tesbed ポータルサイトのスクリーンショット

4.2　ポータルサイト

図 9 は，PIAX Testbed ポータルサイトのスクリーンショットである．

すべての操作は，ウェブブラウザ上で行なうことができる．利用者は，ポータルサイトにログイン (a) すると，割り当てられたノードが表示されるピア割り当て画面となる (b)．この画面で，各ノードに何個づつピアを割り当てるかを設定する．

パッケージは利用者ごとに何個でもアップロードすることができる．アップロードされたパッケージのうち，いずれを起動するか等は，パッケージ操作画面 (c) 上で行なう．所望のパッケージの「起動」ボタンを押すことで，エージェントを分散ピア上で起動させられる．また，各パッケージは「共有化」することができる．この設定をすると，他の利用者が，その利用者自身のパッケージ操作画面にインポートし自分のパッケージ同様実行することができるようになる．また，本稿では詳細に触れないが，API キーを払い出せば，他のウェブアプリケーションやブラウザと，RESTful API 経由で連携動作させることもできる．

パッケージのピア上での起動が完了すると，(d) エージェント操作画面となる．この画面で，エージェントに対する遠隔メソッド呼び出し，発見型メソッド呼び出しを実行する．メソッドに対する引数を指定して「呼出実行」ボタンを押せば，指定されたピア上の MainAgent，または，指定された条件を満たすエージェント上のメソッドを呼び出せる．メソッド実行の返り値は，画面下部のテキストボックスに表示される．

また，エージェントに対する操作は，エージェント間の呼び出し関係 (e)，エージェントが出力するログ (f) を，リアルタイムに収集・表示しながら実行することができる．

これらのポータルサイトの機能によって，遠隔での複数ノード上での実行であっても，ローカルの実行環境と遜色ない実行環境が実現されている．

5.　おわりに

本稿では，我々が大量のノード上での分散アプリケーションの実証実験を簡単化し，少ない手間で検証することを可能とするテストベッドとして開発してきた PIAX Testbed の設計と実装について述べた．

PIAX Testbed は，現在，JGN-X の複数拠点上で運用中であるが，今後，利用可能な計算機リソースの大幅な増強や，センサーネットワーク等の小さなデバイスを含む異種環境にも対応できるよう改良していく予定である．

参考文献

[1] Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M. and Bowman, M.: *PlanetLab: an overlay testbed for broad-coverage services*, ACM SIGCOMM Computer Communication Review, Vol.33, Issue 3, pp.3-12 (2003).

[2] Nakao, A., Ozaki, R. and Nishida Y.: *CoreLab: An Emerging Network Testbed Employing Hosted Virtual Machine Monitor*, In Proc. of the 2008 ACM CoNEXT Conference, pp.73:1-73:6 (2008).

[3] 独立行政法人情報通信研究機構：JGN-X の PIAX テストベッド (online) 入手先 ⟨http://piax.jgn-x.jp/⟩ (2013.04.08).

[4] 独立行政法人情報通信研究機構：新世代通信網テストベッド JGN-X (online) 入手先 ⟨http://www.jgn.nict.go.jp/⟩ (2013.06.28).

[5] 吉田 幹, 奥田 剛, 寺西 裕一, 春本 要, 下條 真司：マルチオーバレイと分散エージェントの機構を統合した P2P プラットフォーム PIAX, 情報処理学会論文誌, Vol.49, No.1, pp.402-413 (2008).

[6] PlanetLab experiment manager (online): 入手先 ⟨http://www.cs.washington.edu/research/networking/cplane/⟩ (2007.08.16).

[7] pShell (online): 入手先 ⟨http://cgi.cs.mcgill.ca/%7Eanrl/projects/pShell/index.php⟩ (2007.05.14).

[8] pssh (online): 入手先 ⟨http://www.theether.org/pssh/⟩ (2009.10.21).

[9] 首藤 一幸, 田中 良夫, 関口 智嗣：オーバレイ構築ツールキット Overlay Weaver, 情報処理学会論文誌: コンピューティングシステム, Vol.47, No.SIG12 (ACS 15), pp.358-367 (2006).

[10] 梅津 高朗, 池田 直徒, 東野 輝夫：P2P アプリケーションの開発と性能評価のための統合開発環境の提案, 情報処理学会論文誌, Vol.47, No.7, pp.2194-2201 (2006).

[11] Aspnes, J., and Shah, G.: *Skip graphs*, ACM Trans. Algorithms, Vol. 3, 4, Article 37, 2007.

[12] 小西 佑治, 吉田 幹, 竹内 亨, 寺西 裕一, 春本 要, 下條 真司：単一ノードに複数キーを保持可能とする Skip Graph 拡張, 情報処理学会論文誌, Vol.49, No. 9, pp.3223-3233 (2008).

[13] Yuuichi Teranishi: *PIAX: Toward a Framework for Sensor Overlay Network*, in Proceedings of 6th Annual IEEE Consumer Communications & Networking Conference(CCNC 2009 Workshops), pp.1-5 (2009).