分散システムの開発を支援するテストベッド
—数台の計算機で大規模テスト・開発環境構築—

1. 背景

近年、高速なネットワークの普及により、P2P システムなどの大規模分散システムが数多く開発されるようになってきた。しかし従来のクライアント・サーバモデルが一対一で通信が行われるのに対し、これらのシステムでは常に数多くのマシンが相互に通信を行っている。このような分散システムの開発は容易ではない。なぜなら、多数のマシンが相互に通信し正常に協調動作をすることを検証しなければならないからである。

このため、P2P などの分散システムのテスト・検証を行うには、多数の計算機が必要となる。しかし、開発者がテスト環境構築のために実機を用意することは、コストが大きい。この問題に対しては、近年発展した仮想マシンを用いて、仮想的に計算機を用意するという手法が考えられる。しかし、既存の仮想マシンの多くは大量の資源を消費するため、1 台の計算機上に実現できる実行環境が、高々数個から数十個に限られるという問題がある。またテスト環境を用意でき実行できたとしても、各ノード・システム全体の状況を把握しにくいという問題もある。

2. 目的

本プロジェクトの目的は 1 台または数台の計算機を用いて数百から数千の仮想環境を生成することで分散システムのテストを可能とし、加えて様々な仕組みによりテスト・開発の支援を行うことのできるミドルウェアを開発することである。本ミドルウェアにより、大規模なテスト環境を持たない開発者であっても、容易にテスト環境を構築することを可能とさせる。またグラフ、アニメーション、デバッグツールなどを用いて、通常困難な分散実行における各ノード・システム全体の状況の把握を容易にさせる。これにより、開発者がシステムの動作の正当性、パラメータの変化による実行の変化を検証することが可能となる。

3. 開発の内容

本ミドルウェアは Linux 上に実装されており、x86・x86-64 アーキテクチャにおいて動作する。実装は全てユーザレベルで行っており、オペレーティングシステムやアプリケーションの修正、管理者権限は不要である。このため、ユーザが共用の計算機などでも容易に使用することが可能であり、実機の用意という面でもユーザの負担を軽減する。

使用の流れとしては、まずユーザが仮想環境の定義やテストの実行手順（シナリオ）を記述する。それを基に本ミドルウェアが仮想分散実行環境を構築し、テストを自動的に実行する。テスト中は、デバッグツールを用いて効率的にデバッグ作業を行うことができ、またアニメーションにより各ノードやネットワークの状態を確認することができる。テスト後は、本ミドルウェアが記録したログ情報からテスト結果をグラフ、アニメーション、テキストを用いて表示し、ユーザは効率的にテストの解析を行うことが可能となる。

本ミドルウェアの開発は、複数の計算機上での多数の仮想実行環境の生成と大規模分散システムのテスト・開発を支援する機構の大きく二つに分けられる。

3. 1 仮想実行環境の定義・生成

仮想環境の定義は自動定義やループ構造を使用することができ、少ない記述で多くを定義することが可能である。仮想環境の生成に関しては、従来の仮想マシンなどでは、本プロジェクトの目的とする規模の仮想分散実行環境を生成することは非常に困難であ

る。そのため、プロセスレベルの仮想化技術により、アプリケーションが必要とする資源に対してのみ仮想化を行うことで、多くの仮想環境を高速に動作させた。具体的には、Linux のプロセストレース用 API である ptrace システムコールを用いて、システムコールの引数を書き換えることなどにより、各アプリケーションに独自のファイル空間とネットワークを仮想的に提供する。それにより、20 台によるクラスタを用いて 6000 の仮想環境を構築することができ、各仮想環境上でアプリケーションがストレスなく正常に動作した。テストは Gtk-Gnutella、Mutella、DBHub、microdc2 などの多くの P2P アプリケーションを用いて行い、全て正常に動作することを確認した。仮想環境上で動作する Mutella を図 1 に示す。

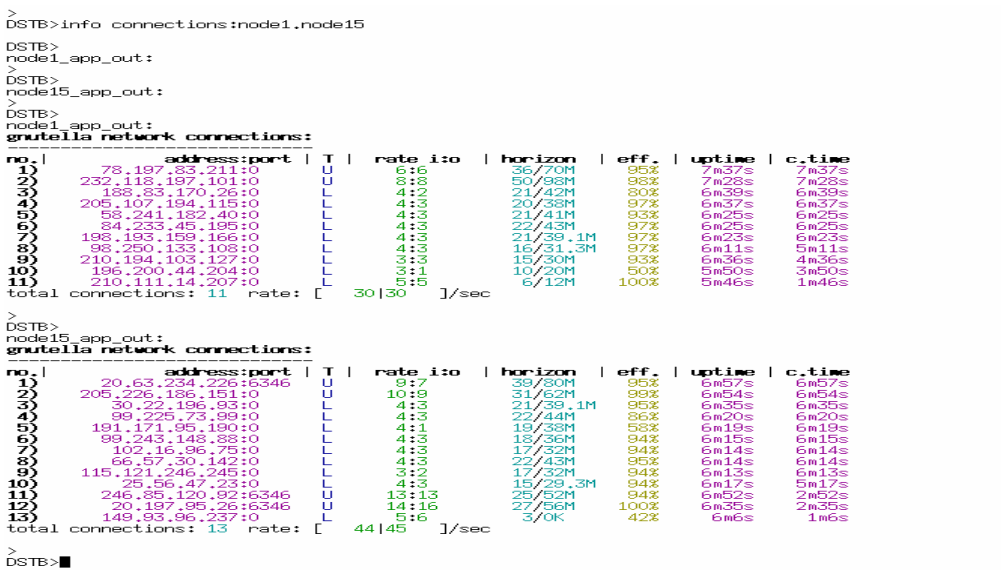

図 1 仮想環境上で動作する Mutella

3．2 テスト・開発を支援する機構

テスト・開発を支援する機構としては、ネットワークエミュレーション機能、テストの自動化、仮想環境の状態を提示する GUI、デバッグ支援機構、テスト結果解析機構などを実装した。

・ネットワークエミュレーション機能

ネットワーク状態に応じた動作検証を行えるようにするため、NAT、リンク間の遅延のエミュレーションを行えるようにした。これにより、分散システムにおける NAT の問題、リンク間の遅延によるアプリケーションのノード形成への影響を検証することが可能となる。

・テストの自動化

ユーザが事前に記述したシナリオに従い、テストを自動的に行えるようにした。数千規模の仮想環境のテストにおいて各ノードにコマンドを実行させることは労力が大きく、入力ミスなども起こりえる。また同じテストを再現することは難しい。テストを自動化することにより、そういった開発者の労力を軽減し、再現性を持たせることを可能とさせる。

・仮想環境の状態を提示する GUI

各ノードやネットワークの状態・通信状況をアニメーションにより表示する。これにより、仮想環境の状況を直観的に理解することができ、デバッグや実験結果の解析が効率的

になる。図 2 に 50 の仮想環境を構築し、P2P アプリケーションを動作させた時の GUI のスナップショットを示す。

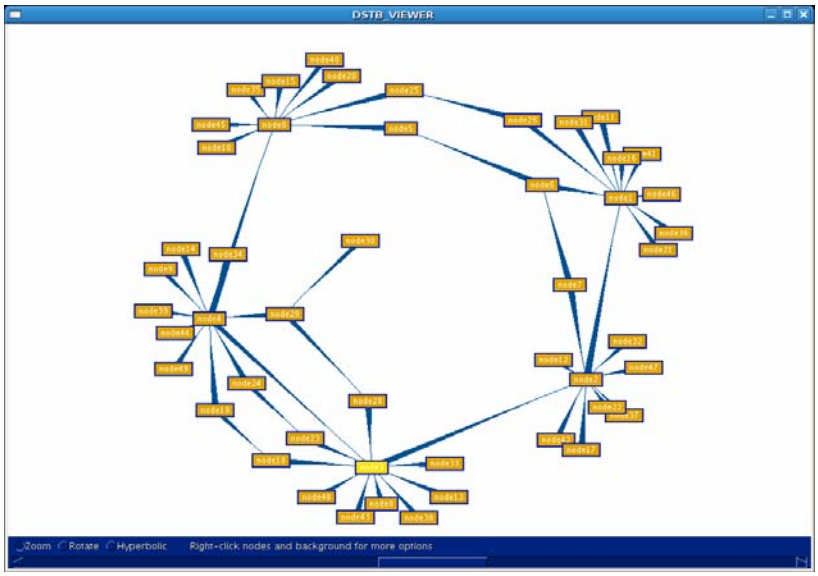

図 2 仮想環境の状態を提示する GUI

・テスト結果解析機構

分散システムにおいてログの解析は非常に複雑であり時間のかかる作業である。そういった開発者の負担を本機構により軽減する。テスト中のノードやネットワーク状態・通信状況のログを記録し、その情報をグラフやアニメーション、テキストを用いて表示を行う。これにより、開発者がシステムの実行、ネットワーク構築が正常かつ適当に行われているかどうか検証することを支援する。本機構のグラフによる表示例を図 3 に示す。

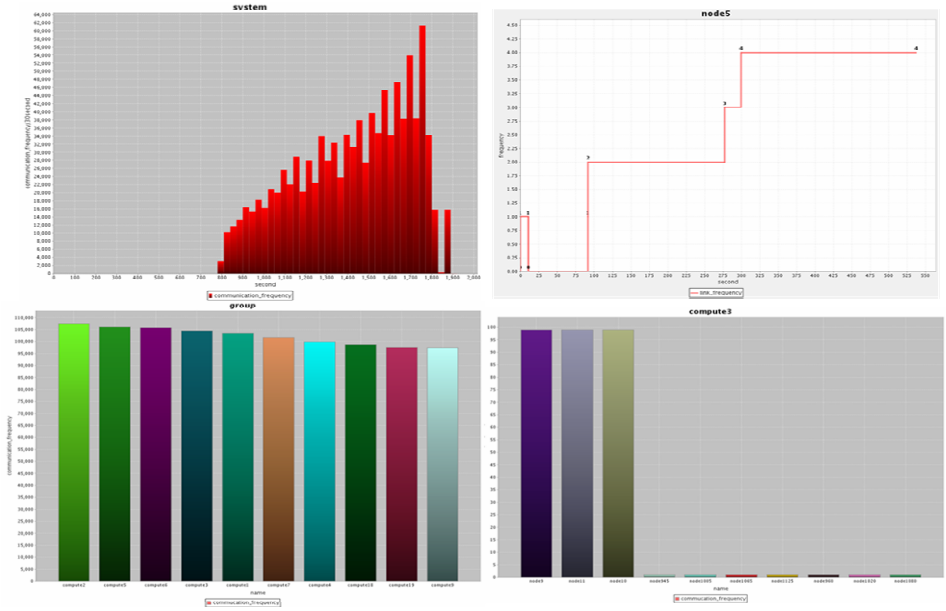

図 3 テスト結果解析機構のグラフによる表示例

・デバッグ支援機構

複数のノードを同時に実行している場合、そのデバッグ作業の労力は大きい。それぞれ

のノードに対してデバッガを起動し操作を行わなければならない。また他ノードとの接続・通信状況をプロセスとノードの対応を調べながら行わなければならない。そういった労力を軽減させるため、テスト中に仮想環境上のアプリケーションの挙動が疑わしい、メモリアクセスエラーなどでダウンした場合に、効率的にデバッグ作業を行える機構を実装した。この機構は GDB を用いて、一つの端末（kterm、 xterm など）から複数の仮想ノード(プロセスではなく)に対してデバッグ作業を行うことが可能となる。また、アプリケーションへの入力・出力の確認についても同様に一つの端末から行うことができる。

4. 従来の技術（または機能）との相違

現在分散システムのテスト環境としては、PlanetLab、Netbed、StarBED などのテストベッドや ns-2、GloMoSim、OMNeT++などのネットワークシミュレータがある。テストベッドは、リソースの提供が必要などの条件がある場合もあり、手軽には利用できない。ネットワークシミュレータは、多くの場合テスト専用のコードを記述してシミュレーションを行うものであり、別途実環境で動作するソフトウェアを実装しなければならない。また、実際にアプリケーションを実行してテストを行うものではない。

分散システムの開発において、少ないノード数によるテストでは安定して動作していたが、多くのノード数で大規模なテストを行うと問題が生じるといったケースが多々ある。しかし、実際にはアプリケーションを数千規模で動作させる環境を用意することは非常に難しく、テストベッドにおいても頻繁に数千規模のノードを使用することは難しい。本ミドルウェアは現実世界のネットワーク状態を反映した正確な実行性能の測定や検証などには不向きではある。しかし、テストベッドやネットワークシミュレータを利用したとしても、必ずしも正確な測定や検証が行える保障はなく、運用してみなければわからない面も多い。そのため、本ミドルウェアは少ない計算機で、できるかぎり多くの仮想環境を生成することで大規模のテスト環境を構築し、開発を支援する機構により動作検証を正確に行えることを目的としている。

5. 期待される効果

現在分散システムの開発を行う場合、特に個人ではそのためのテスト・開発環境を構築することは非常に困難である。1台もしくは数台で数百から数千規模の仮想環境を構築し、開発の支援を行うミドルウェアを開発することにより、個人での分散システムの開発にも貢献できる。それによって、一般にも多くの分散システムが開発され、より身近のものになっていくことが期待される。

6. 普及（または活用）の見通し

開発したミドルウェアは前章で述べた通り、少ない計算機で大規模なテスト・開発環境を構築できる。またオペレーティングシステムやアプリケーションの修正、管理者権限は不要であり、利用が容易である。そのため、企業や大学に限らず、個人も含めた多くの開発者に使用していただきたいと考えている。今後、下記の（参考）開発者 URL においてフリーソフトとして公開する予定である。

7. 開発者名（所属）

西川　賀樹（東京大学大学院　情報理工学系研究科　博士課程 1 年）

（参考）開発者URL

http://www.yl.is.s.u-tokyo.ac.jp/~zbkt/