分散プログラミングモデル：Federated Linda

Distributed Programming Model : Federated Linda

048545E　安村 恭一　　指導教官:　河野 真治

1　はじめに

比較的ネットワーク的に遠いコンピュータを取り扱う分散プログラムはスケーラブルに書くことが難しいとされている。一方、逐次プログラムなどは構造型やオブジェクト指向などにより、難しさが緩和されており、自然に逐次プログラムを書くことができる。これまでの分散フレームワークやツール群は、逐次プログラミングモデルの延長だったりスケーラブルに書くには複雑で理解しづらいものであり、真の分散プログラミングモデルとはいえない。分散プログラムには自然にスケーラブルな分散プログラムを書くことができる真の分散プログラミングモデルが必要である。

本研究の目的は、スケーラブルな分散プログラムを自然に記述できる、真の分散プログラミングモデルを提供することである。本論文では、本研究室で開発した非同期型 Linda の拡張を用いて、分散プログラミングモデル Federated Linda を提案する。

2　分散プログラム

分散プログラムが難しいのは、それをスケーラブルにすることが難しいからである。実際、Internet 上でも、集中サーバ構成が通常であり、きちんとスケールする分散アプリケーションは珍しい。しかし、DNS のように大規模の対象のサービスもあることから、きちんと分散アプリケーションを作ることができればインターネットのような大規模な対象のサービスも実現可能といえる。

自然な分散プログラミングを目指すためには、以下のようなことが要請される。

- 実際の通信のモデルが理解しやすい
- Basic のように会話的に開発できる
- 基本オペレーションが少なく、明解である
- 分散環境での運用が容易
- インターネット環境で自動的に相互接続する

自然に書いて、スケールする分散プログラムになることが目標である。そのためには分散アルゴリズム自体を内蔵するようなプログラミングモデルが望ましい。

2.1　Linda

本論文で提供する Federated Linda のベースとなった分散システム Linda について説明する。Linda はタプルという ID で番号づけられたデータの塊を in,read,out などのオペレーションを用いて、共有されたタプル空間に出し入れすることで分散プログラムを行う (図**??**参照)。Linda の利点として通信モデルが理解しやすいことが挙げられる。また、接続切断に強いモデルである。しかし、集中型になりやすいモデルである。

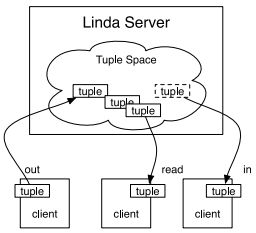

図 1: Linda Server

2.2　分散プログラムの要素

分散プログラムには三つの要素で構成される。Protocol Engine,Local access to protocol, Link configuration である。Protocol Engine はプロトコルを定義している要素である。Local Access to protocol は直接通信にアクセスする API である。Link Configuration は論理的な接続を規定・接続を行う。これらの要素を分離してサポートすることで、プログラムの開発・テストで、それぞれに集中することができる。また、各要素毎に切り換えを行うことができるので、ポータビリティ性の向上が期待できる。

3　Federated Linda の提案

Federated Linda は、複数のタプルスペースを相互に接続することにより分散プログラムを実現する。一つのタプルスペースには少数の接続があることが期待されており、多数のタプルスペースが接続により分散アプリケーションを実現する (図**??**参照)。インターネットのパケット転送のように、タプルと呼ばれる ID とデータが組になったものが、タプルスペース間を転送されていく。

Federated Linda は分散プログラムの要素毎に以下のモデルを提供する。Local Access は、in,read,out を用いたタプルを出し入れする通信モデルであり、通信は非同期に行われる。Protocol Engine は分散アルゴリズムを内蔵するエージェントを記述するモデル。エージェントはタプルをリレー転送する Link Configuration は XML でトポロジを定義して、それを基に接続を行うモジュールを提供する。

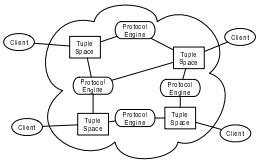

図 2: タプルスペースの相互接続

4 プログラミング例

Linda Server は C 言語で記述している。タプルスペースはメモリ上のキューとして実装されている。タプル ID は 16bit 整数値、データは任意の文字列である。Protocol Engine，クライアントプログラムは C，Perl，Python，Ruby で記述可能である。以下に Python で記述したマルチキャストを行うプログラム例を載せる。

```
while (1) : # メインループ
  rep=ConnectRep.reply()
  if (rep):     # 新規の接続処理
    ConnectRep = self.linda.In(TUPLE_ID_CONNECT)
    # 新規接続者を登録
    self.tuplespaces.append(getTuplespace(rep))
  rep=MultiCastRep.reply()
  if (rep):     # マルチキャスト
    MultiCastRep=self.linda.In(TUPLE_ID_MULTICAST)
    # 全タプルスペースへデータを送信
    for t in self.tuplespaces :
      t.Out(TUPLE_ID_RECV, rep)
  self.flinda.sync()
```

5 Federated Linda を用いたプログラミング

Federated Linda を用いたプログラム例としてディスタンスベクタ型ルーティングプロトコルを Python で実装した。宛先情報は ‘‘ホスト名:ポート番号’’ の文字列として表す。Linda Server とルーティングエージェントをセットで 1 ノードとしてルーティングを行う。今回接続を行ってからルーティングテーブルが安定するまでの時間を計測した (図??)。トポロジは二分木とメッシュを使用した。また、トポロジを構築する XML を用いて、ノード間の接続を行った。この結果より、ディスタンスベクタ型ルーティングプロトコルでは二分木よりメッシュの方が遥かに時間がかかることが容易に確認できた。

6 比較

JXTA はノードの発見とグループの自己組織化は提供するが、分散アプリケーションをどう書くかは示してくれない。Federated Linda はタプルの出し入れという通信モデルと分散プログラム毎にプログラミングモデルを提供している。

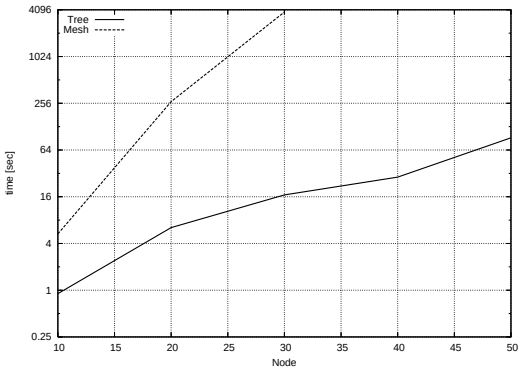

図 3: テーブルが安定するまでの時間

CORBA はネットワークを意識しないオブジェクト呼び出しができるが、それゆえ通信が起こるタイミングが把握しづらくなりやすい。Federated Linda は sync という関数でしか通信は起きないので通信発生は明確である。

OverlayWeaver は分散アルゴリズムを Java でしか記述できない。Federated Linda は好きなスクリプト言語で記述可能である。また、通信モデルも理解しやすい。

7 考察

タプルの出し入れという通信モデルは直感的に分かりやすく、簡潔に書ける。また、分散プログラムを要素毎に分離して提供するので、開発しやすくなっている。Federated Linda の記述の容易さはそれらの理由によるものであると考えられる。タプルの出し入れという通信モデルにより、タプル転送を行う。この通信モデルはインターネットのパケット転送と酷似しており、Federated Linda を用いてインターネットの分散プログラムが記述できる。また、分散プログラムの要素を分離して提供しているので、それらの切り替えが容易に行うことができる。

8 まとめと課題

スケーラブルな分散プログラミングモデルとして Federated Linda を提案し、実装した。またそれらを用いたルーティングプログラムを作成した。他の分散プログラムツールとの比較を行い、考察を行った。今後の課題として、Federated Linda を用いてより多くの分散プログラムを実装することが重要だと思われる。

参考文献

[1] 河野 真治，仲宗根 雅臣，同期型タプル通信を用いたマルチユーザ Playstation ゲームシステム，卒業論文，1998

[2] 安村 恭一，河野 真治，大域 ID を持たない連邦型タプルスペース Federated Linda，第 99 回 情報処理学会 システムソフトウェアとオペレーティング・システム研究発表会，2005.

[3] 安村 恭一，河野 真治，，動的ルーティングによりタプル配信を行なう分散タプルスペース Federated Linda，日本ソフトウェア科学会第 22 回大会，2005.