# JavaSpacesによる性能評価と共有空間の階層化

富田 昌平 大谷 真 坂下 善彦
湘南工科大学 工学部 情報工学科

## 1. はじめに

分散共有メモリ資源を操作するための汎用的なインタフェースを提供するLindaシステム[3]がある．そこでは共有メモリ機構を基盤として，ネットワーク環境上で利用することが可能で，ユーザはオブジェクトがどこに存在するのかを意識せずに位置フリーに扱うことができる．そこでは，抽象的な操作インタフェース“Linda Interface”が用意されている．共有メモリ上にtupleによる共有空間を提供，この共有空間にオブジェクトを置くことができ、このオブジェクトの操作は，書込み、読込み、取出しの３つの操作を基本により行う．この原理をJava上で使えるようにしたのがJavaSpacesである．JavaSpacesではtupleのことをEntry、共有空間のことをJavaSpaceと呼ぶ.JavaSpacesはJiniによって実装される．Jiniはモジュール協調サービスの形をとる分散システムの構築のためのネットワークアーキテクチャを提供する．

これによってOSなどの異なるコンピュータ資源で繋がっているネットワーク上でも共有空間を作り出すことができ、他のコンピュータに存在するデータ・資源などのオブジェクトを論理的に共有することが可能となる．

我々は、この共有空間を構造化させることにより，さまざまな応用に適用することが出来ると考えている[1][2][4]．

本研究では，空間を分割する方法を検討し，分割管理することによる操作上のオーバーヘッドの量を計測した．

次に，分割の方法を排他的あるいは平面的に分割するのみならず，それぞれの分割領域が互いに重なる関係を保つことの出来る管理の方式を検討した．

## 2. 階層化の方法

### 2.1 階層化の考え方

対象としているJavaSpacesは，Entryによって構成されている．即ち、Entryされているオブジェクトだけで構成されている単純なものとなる．

Structured common memory system based on JavaSpaces and the evaluation
Shohei Tomita, Makoto Oya, Yoshihiko Sakashita
Information Science, Faculty of Engineering,
Shonan Institute of Technology

このEntryの一部を階層化の目的に沿ったデータ形式として定義し，階層構造の管理制御の目的に使用する．この構図を図１に示す．

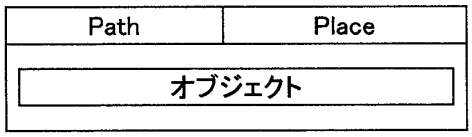


図１．構造化データを付加したオブジェクト

また，構造を管理するモジュールが必要となるが，ユーザ側が操作を行うあらゆる場合にこの管理モジュールに問い合せたり，実行を依頼する手法では明らかにオーバーヘッドが増大してしまう．また，折角JiniあるいはJavaSpacesが一透過性を提供している特徴をつぶしてしまうことになる．この理由で，その時点での構造あるいは互いの関係をユーザ側で独自に判断できることを前提とした．

### 2.2 構造の管理制御

ユーザからの要求内容をJavaSpace内にEntryする，この要求を管理制御エージェントがJavaSpace内から要求を拾い出して，構造の管理制御を行う，その結果を再度JavaSpace内に置き，この置かれた結果を要求したユーザエージェントが知ることで，図２に示すように一連の処理操作が完了する．

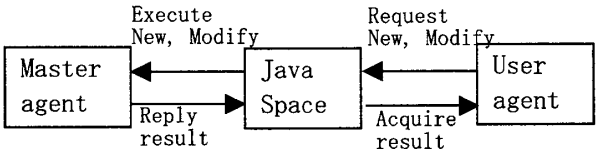


図２．エージェントとJavaSpaceのやり取り

各種構造の構成要素は，管理制御エージェントがJavaSpace内のEntryの関係情報を読み込むことにより，ユーザはJavaSpaces内に存在する構成要素の情報から，Entryの位置関係を知ることが出来る．

### 2.3 構造への入力操作

ユーザは構造へ入力データをJavaSpace内に書込む際、構造内の目的の場所情報を付加する．この要求を管理制御エージェントが受け、ユー

ザから入力されたデータを取出す. 管理制御エージェントは入力された場所が同じ Entry を探し出し、Entry につけられている path 情報を読み込む. 取出した path 情報と入力された場所情報をユーザから入力されたデータに付加して、管理制御エージェントが JavaSpace 内に書込むことで、入力操作は終了する.

## 3．構造管理の処理時間の測定方法

2台の Pc により JavaSpaces を構成し、処理時間を測定する．今回使用した Pc スペックは表 1 に示す Pc1 に JiniServer を、Pc2 に管理制御エージェントを配置してデータ入力を行うものと、Pc2 に JiniServer と管理制御エージェントを配置してデータ入力を行うものの二つの方法で測定する．これにより、ネットワークを介して JiniServer を利用した時のオーバーヘッドが分かる。測定の仕方は以下のとおりである．まず、木構造の Entry を作っておきそれを元に、管理制御エージェントを使ってデータを適当な Node と path を付加して JavaSpace 内に書き込む処理と、管理制御エージェントを使わず JavaSpace 内に書き込む処理を行い、この処理を 5,000 回繰り返し行ったときの処理時間を測定する．今回測定するために作った木構造の Entry 内容を表 2 に、そのモデルを図 3 に示す.

表 1．今回使用した Pc スペック

| | OS | CPU | メモリ |
|---|---|---|---|
| Pc1 | WindowsXP | Celeron, 2.53GHz | 1 GB |
| Pc2 | WindowsXP | Core2, 1.8GHz | 1 GB |

表 2．処理時間測定の Entry

| path | Place | データ |
|---|---|---|
| ROOT/ | A | ObjectA |
| ROOT/A/ | A1 | ObjectA1 |
| ROOT/A/A01/ | A01-1 | ObjectA1-1 |
| ROOT/A/A01/ | A01-2 | ObjectA1-2 |

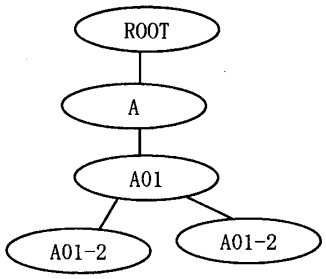


図 3．処理時間測定の木構造

## 4．測定結果

JiniServer と別でデータ取った方の管理制御エージェントが書き込む処理の 1 データあたりの平均処理時間は 4.75ms で、管理制御エージェントを使わずに書き込む処理の 1 データあたりの平均処理時間は 1.33ms であった.

1台でデータを取った方の管理制御エージェントが書き込む処理の 1 データあたりの平均処理時間は 2.28ms で、管理制御エージェントを使わずに書き込む処理の 1 データあたりの平均処理時間は 0.30ms であった．これにより、管理制御エージェントを使った時のオーバーヘッドとネットワークを介して JiniServer を利用した時のオーバーヘッドが出た．その結果を表 3 に示す.

表 3．オーバーヘッド

| | |
|---|---|
| 1台での Agent 有無のオーバーヘッド | 2.0ms |
| 2台での Agent 有無のオーバーヘッド | 3.4ms |
| JiniServer をネット仲介した時のオーバーヘッド | 1.4ms |

## 5．まとめ

分散処理環境 Jini により制御される共有空間を構造化し、構造を管理制御する仕組みを構築した．構造を管理制御するオブジェクトの動作時間は、簡単な構造では 2～3.5mSec のオーバヘッドに収まった．大きな構造あるいは複雑な構造に対する測定評価が必要である.

今回用いた Jini がネットワークを越えて対象の tuple を操作する場合は、約 1.4mSec ほどの時間を要していることが分かった．これは、ネットワーク上に大きなあるいは広い共有空間を対象とした場合には更に検討が要る.

参考文献

[1] 坂下, 稲守：ネットワーク“場”における情報収集モデル, 情報処理学会マルチメディア・分散・協調とモーバイル(DiCoMo)シンポジューム, pp. 181-184, 2003

[2] 倉前宏行、島野顕継、木村彰徳、松本政秀、古野良樹、矢島, 坂下: 情報場を備えた JavaSpaces による情報共有空間の制御, 情報処理学会第 68 回全国大会, 2006. 3

[3] Sudhir Ahuja, Nicholas Carriero, David Gelernter: Linda and Friends, IEEE COMPUTER, pp. 26 34, 1986

[4] 坂下、大谷、富田：分散環境内を移動する実行主体の構築—情報場と Web エージェント—、電子情報通信学会　信学技法、Vol. 107 No. 366, pp. 49-55, 2007