M-050

分散共有メモリ JavaSpaces を階層構造化した情報場の管理と操作
Control and operation to the proposed Layered Information Space based on distributed shared memory:JavaSpaces

富田 昌平† 坂下 善彦‡ 大谷 真‡
Shohei Tomita† Yoshihito Sakashita‡ Oya Makoto‡

1. はじめに

近年コンピュータ技術の発展によりコンピュータの性能やネットワークの性能は飛躍的に向上してきている。このため，計算量の多い大規模計算には複数台のコンピュータをネットワークで結んで分散して計算する分散コンピューティングの技術が注目されている。この技術の一つに JavaSpaces[1]がある。これは複数台のコンピュータのメモリ資源を集め，一つの情報の共有空間（タプルスペース）を作り出す技術である。この共有空間に入れられる情報を今回はオブジェクトと呼ぶ。

本研究では JavaSpaces によって構成される共有空間に入れるオブジェクトをグループ別に分けて構造化する手法を提案する。

2. JavaSpaces と情報場

2.1 JavaSpaces

JavaSpaces は，オブジェクトを共有する空間環境を提供する。また，分散処理環境を提供するシステムである Jini[2] のサービスの１つである。即ち，図１の様に JavaSpaces で作られた共有空間は Jini によって分散処理環境に実装される。

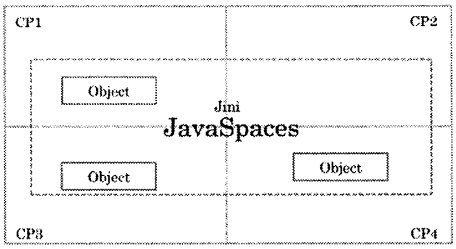

図１ 共有空間の実装

2.2 情報場と階層化

本来，JavaSpaces によって作られる共有空間は１つであり，全てのオブジェクトを誰でも見ることができ，接続することができる。この利点は，誰でも共有空間の中のオブジェクトを利用することができ，自由にオブジェクトを提供することができる点である。しかし，オブジェクトの提供者が多くなり，共有空間の巨大化した場合，必要なオブジェクトを見つけるのには時間がかかるようになる。また，必要なオブジェクトは１つとは限らない。複数のオブジェクトを１つのグループとして扱ことによってより利用価値が増えると考える。そこで我々は共有空間に入るオブジェクトにグループ化に必要な付加情報をつけることによって共有空間にあるオブジェクトをグループ化させる研究を行っている。我々はこのグループを「情報場」[3]と呼ぶ。更に，図２に示すように作成された情報場同士を階層的関係に構築するシステム[4]を提案する。

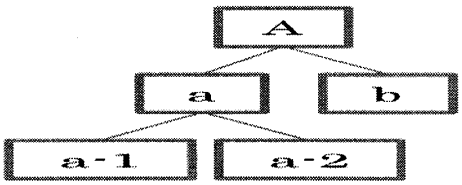

図２ 階層構造を持つ情報場

図２は情報場 a-1 と a-2 が存在し，この二つの情報場を元に情報場 a が成り立つ。これと情報場 b を合わせることによって全体の情報場 A が成り立っている図である。

これにより共有空間を利用する際に，使用するオブジェクトを情報場単位で扱うことができ，これらの情報場が階層的関係を持つことによって，図２の例では通常 a の単位でオブジェクトを扱っていた場合，取り扱う範囲を広げたければ A，狭くしたければ a-1 などと使用するオブジェクトの適用範囲を意図に合わせて限定したり，広げたりすることが可能になる。

3. 階層構造の管理制御

3.1 システム概要

本システムは図３のような振る舞いをするシステムである。

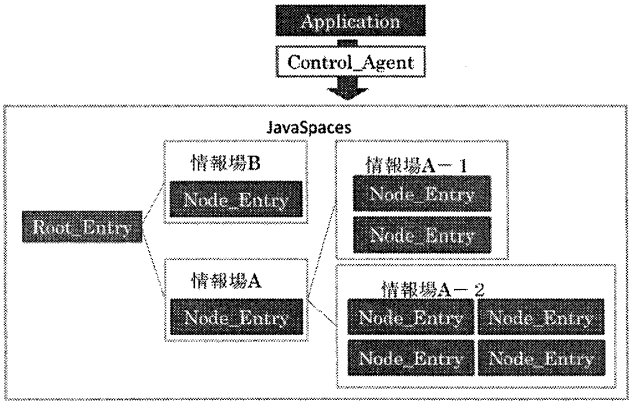

図３ 階層構造化された情報場

図が示すように Application は Control_Agent を通して JavaSpaces に接続することによって Root_Entry と Node_Entry を利用することによってあたかも情報場を形成し，構造化されているように見える。実際のシステム構成を図４に示す。

†湘南工科大学工学研究科
Shonan Institue of Technology, Graduated School
‡湘南工科大学情報工学科
Shonan Institue of Technology, Dept. of Information Science

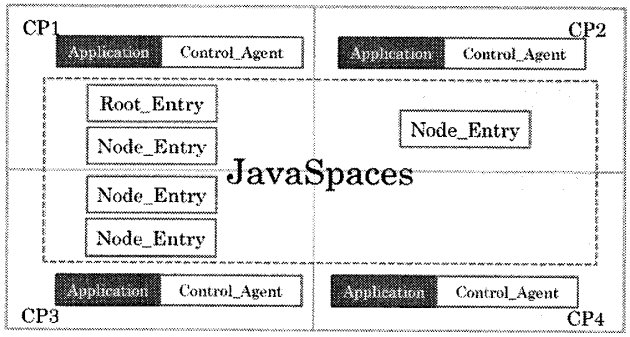

図４　システム構成概要

オブジェクトは Node_Entry に入れられて共有空間に置かれる。Node_Entry は「Path 情報」を持っている。これはオブジェクトが置かれている情報場を記録している情報である。Root_Entry に全ての Path 情報を保存することによって，Root_Entry に問い合わせることで情報場の階層構造を知ることができる。次にシステムの主要となる部分を説明する。

3.2 Control_Agent

Control_Agent にはユーザやアプリケーション側に Root_Entry の Path 情報から階層構造情報を掲示する機能や，Node_Entry の書込みや読込みなどの操作することによってオブジェクトを情報場として扱い，それを階層構造化させるための機能などがある。これらの機能によって情報場の生成と階層構造化が行われている。

3.3 Node_Entry

Node_Entry とは情報場を生成するための共有空間にオブジェクトを置く入れ物である。これはオブジェクトと自身の Path 情報，自身より一つ下位の Node_Entry の Path 情報を持つ。これは自身の Path 情報を上位の Node_Entry もしくは Root_Entry に伝える機能がある。

3.4 情報場の生成

情報場が生成された共有空間を図５に示す。

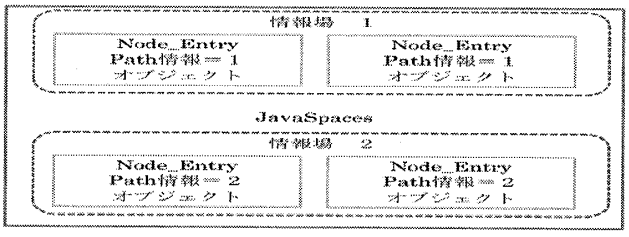

図５　情報場の生成

ユーザやアプリケーションは Control_Agent を通してオブジェクトを共有空間に書込こむ際に Path 情報として自身の位置情報を書込む。これによって情報場が作られる。

3.5 Root_Entry

Root_Entry とは全ての Node_Entry の上位に位置し，各 Node_Entry から送られてくる Path 情報を全体の情報場の位置情報として保管する機能を持つ。

3.6 情報場の階層化

階層化された情報場の書込み処理の処理過程を図６に示す。

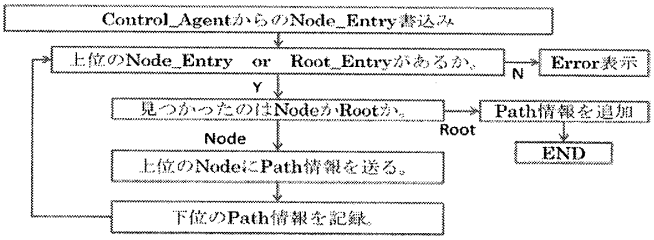

図６　階層化の書込み手順

4. 実装結果

本システムを実装した際に構造に関連する処理のオーバーヘッドが発生した。この結果を図７に示す。

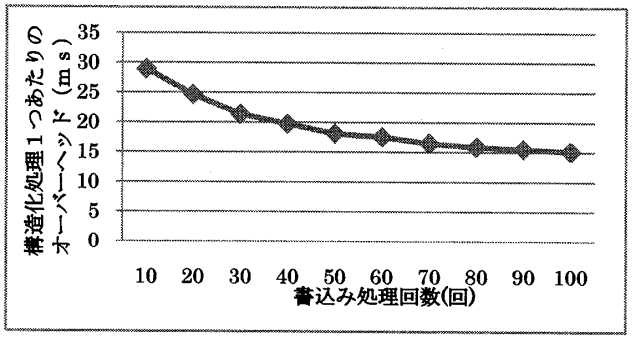

図７　オーバーヘッド

図が示すように書込み処理が増えることによって１処理あたりのオーバーヘッドが減っているのがわかる。

5. 結論

本論では JavaSpaces に入れられるオブジェクトを情報場として扱い，更に情報場を階層的関係に構築する手法を提案し，開発を行った。この手法で情報場を構築し，階層化を行った場合，階層の深さが深くなるほどオブジェクトの書込みに全体の時間がかかるが一つの処理に対する処理時間は短くなることがわかった。原因は似た処理を何度も行っているため，JavaSpaces 内のキャッシュ効果によって１処理あたりのオーバーヘッドが減ったと考えられる。

6. まとめ

今回の手法で情報場の階層構造階層の深さが深くなるほど書込みに時間がかかってしまうので，処理部分を分散処理できるように改良し，JavaSpaces の基本動作として一つのインターフェイスとして実装する計画である。

参考文献

[1] Sun Microsystems:Getting Started With JavaSpaces Technology:Sun Microsystems(online), available from <http://java.sun.com/developer/technicalArticles/tools/JavaSpaces/> (accessed 2007-12-05)
[2] Jan Newmarch: Jan Newmarch(online), available from < http://jan.newmarch.name/ > (accessed 2007-12-05)
[3] 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
[4] 富田，坂下，大谷：JavaSpaces によるオブジェクト共有空間の構造化と性能評価，情報処理学会第 69 回全国大会，2008.3