分散共有メモリ間通信方式のクラスタ型並列処理機構

工藤達矢†　坂下善彦‡

湘南工科大学　大学院工学研究科†　湘南工科大学‡

1．　はじめに

クラスタには、分散メモリ方式と共有メモリ方式がある。MPIを用いた分散メモリ方式は、プロセッサ間のメッセージ通信やデータの分割などをユーザが明示的に記述する必要があるので、プログラミングのコストが高い。そこで、分散共有メモリ型が注目されている。分散共有メモリ型は物理的には分散メモリ方式でも、仮想的にメモリを共有して使うことで従来の分散メモリを意識した並列プログラミングに比べて容易に並列プログラムを作成することができる。本研究では、共有メモリ型のオブジェクト共有空間 JavaSpaces を用いて分散共有メモリ型のクラスタを構築し性能評価を行った。

2．分散共有メモリ

2.1　共有メモリ型と分散メモリ型

共有メモリ型のプロセッサは、計算に参加している全ての計算機上のメモリにアクセスすることができる。一般に高価な専用ハードウェアが必要で分散メモリ型に比べ拡張性が乏しく、計算機クラスタ環境に適応させることが難しいので、専用の並列計算機として実現されていることが多い。共有メモリ型は並列化コンパイラを使用して逐次プログラムを自動並列化できプログラミングが容易である。共有メモリはプロセッサの数が多いとメモリアクセスが競合し、プロセッサ数に比例した性能の向上が得られない。

分散メモリ型はプロセッサと主記憶から構成された計算機をネットワークを介して複数接続し、それぞれの計算機上で走るプログラムは、その計算機上のメモリにのみ読み書きする。他の計算機とはメッセージ通信などの方法でデータのやり取りを行う。データの分散を意識したプログラミングをする必要があり、プログラミングのコストが高い。そこで、論理的には共有メモリ環境を提供しながら、物理的には分散メモリ構成にすることで共有メモリと分散メモリの利点を両方持つ、分散共有メモリ型が注目されている。

Cluster-type parallel processing communicated in distributed shared memories

†Tatsuya Kudou, Shonan Institute of Technology, Graduate school

‡Yoshihiko Sakashita, Shonan Institute of Technology

2.2　MPI

MPIは、分散メモリ型の並列計算機のプロセッサ間のデータ通信を行うためのライブラリである。分散メモリ型の並列計算機はそれぞれの計算機は独立したアドレス空間を持ちメモリを共有しないので、メッセージ通信によりプロセッサ間のデータのやり取りを行い、複数のプロセッサがデータをメッセージとして送受信することで並列計算を行えるようにする。メッセージ通信はプロセスから他のプロセスにデータを明示的に送る方法で効率のよい並列プログラムを書くことができる。

2.3　JavaSpaces

JavaSpaces は、ネットワーク上に型指定されたエントリと呼ばれるオブジェクトを格納する共有空間を提供する Jini のサービスである。Linda モデルを Java で実装したものでありJavaSpaces は共有空間に対して write, read, take, notify の4つの操作を行う。

マスターが仕事エントリを作成し、JavaSpaces に書き込み、ワーカーがそれを取り出して計算を行い、結果を JavaSpaces に返し、結果をマスターが集める。マスタ・ワーカ方式により並列処理を行う。

表1　クラスタの構成

ノード数	4台
CPU	AMD Athlon 64 3500+ 2.2GHz
メモリ	1GB
Jini	jini2.1
MPI	Mpich1.2.7p1

3．分散共有メモリ間通信方式

PC1 は JavaSpaces にエントリオブジェクトを書込む。エントリ情報はすぐ JavaSpaces サービスに通知される。

PC2 は JavaSpaces サービスに読み出すエントリオブジェクトの型を指定する。JavaSpaces サービスは指定されたエントリオブジェクトを見つけ出し、エントリオブジェクトが保存されている PC1 の接続先を PC2 に伝える。そして、PC2 はエントリオブジェクトを読出すことができる。

JavaSpaces はオブジェクト共有空間を用いて互いのデータを送受信を行い、特定の相手を明示して通信する必要はないので、MPI では計算途中のプロセスの増減はできないが、JavaSpaces では計算途中でのプロセスの動的な変化にも柔軟に対応が可能であり、計算ノードの一部がトラブルにより停止した際にも対応できる耐故障性にも優れている。

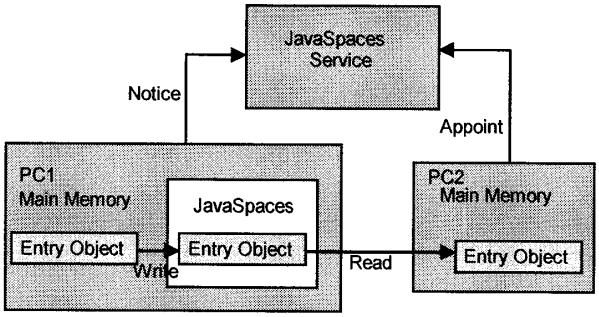

図 1　JavaSpaces 内のオブジェクト通信

4. 性能比較

N-Queen 問題と姫野ベンチを実行し MPI と JavaSpaces の性能の比較をした。

4.1 N-Queen 問題

N-Queen 問題とは N×N サイズの盤面に N 個のクイーンをお互いに攻撃できないような配置を見つける問題である。N-Queen 問題は,代表的な組み合せ最適化問題であり、この問題は問題の定義が簡単でよく知られていることと並列化が比較的容易であることから、並列計算のベンチマークプログラムとしても利用されている。

測定結果を図 1 に示す。 MPI も JavaSpaces を使ったどちらもプロセッサ数に比例した性能の向上が得られた。

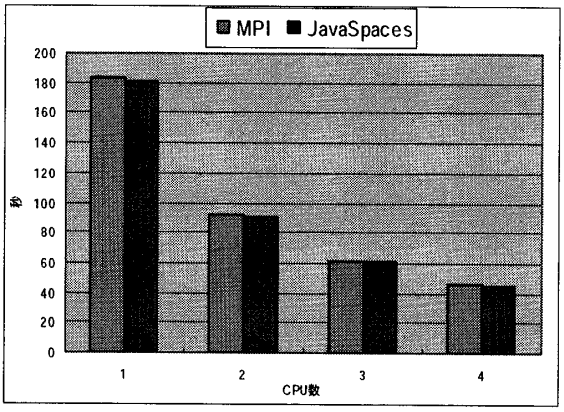

図 2　N=16 の処理時間

4.2 姫野ベンチ

姫野ベンチはポアッソン方程式解法をヤコビの反復法で解く場合に主要なループの処理速度を計るものであり、三次元行列に大量のデータを読み書きすることで、主に計算機のメモリバンド幅などの性能による効果が大きいベンチマークである。並列計算ベンチマークとしてよく使われている。

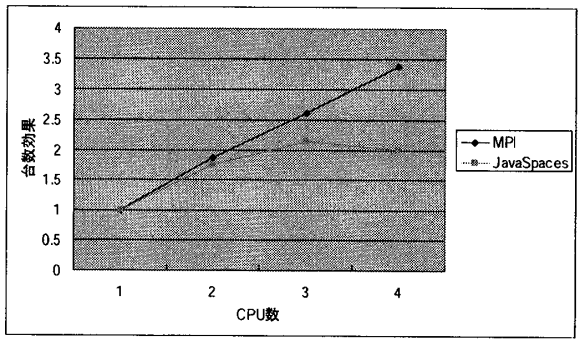

図 3 姫野ベンチ台数効果

姫野ベンチで計測した測定結果を図 3 に示す。姫野ベンチでは、MPI が台数効果が良いのはプロセッサ間でデータを交換する際、MPI は送信側から受信側に直接送るが、JavaSpaces では送信側が共有空間にオブジェクトを書き込み、共有空間から受信側が取り出すため通信に時間が掛かり MPI に比べて通信効率が悪いためである考えられる。

Javaspaces の 4 台での処理速度が低下したのは、Javaspaces では情報量が多い通信が頻繁に発生する場合に Jini のサーバに通信が集中するので、通信効率が低下し全体の性能が低下したと予想される。

5. まとめ

姫野ベンチは計算機の台数を増やすと Javaspaces のサーバに通信が集中し、共有メモリの通信性能が低下した。Javaspaces はこのようなプロセッサ間で情報のやり取りが多い計算では台数効果は得られない。しかし、N-Queen 問題のようなプロセッサ間で送られるデータのサイズも小さく、通信回数も少ない計算では、台数に比例した性能の向上が得られた。

今後の課題としてサーバの負荷を抑える仕組みを検討する。

参考文献

[1] 中山　茂, "Java 分散オブジェクト入門"
[2] 溝渕, 石井, 宮本, 橋本, " Jini 技術を用いた分散並列計算環境" 第 17 回数値流体力学シンポジウム