マイグレーションを支援する分散集合オブジェクト

高橋 慧[†] 田浦 健次朗[†] 近山 隆[†]

1. はじめに

グリッド環境においては、与えられた資源を最大限活用するためにプロセッサ数の増減に対応したいという要請がある。同時に記述が簡単に行え、かつ実行速度を損なわないことが望ましい。現状では多くの並列プログラムはメッセージパッシングを用いて記述されているが、これは下層の処理に近すぎて、簡潔にアルゴリズムを表現できない場合が多い。また多くの場合通信相手の指定にプロセッサ番号を用いるため、プロセッサ数増減に対応するにはデータとプロセッサ番号の対応をプログラマが管理する必要があり、記述は難しい。

これらを改善するモデルとして、分散オブジェクトモデルが提案されている。分散オブジェクトではオブジェクト指向の記述が行え、またデータの識別にプロセッサではなくオブジェクトを用いるので、プロセッサ増減に対応したプログラムの記述が簡単になる。しかし、分散配列などプロセッサ間にまたがる大きなデータを扱う場合、分散オブジェクトではオブジェクト単位でしかデータを扱えないため、簡単には効率の良い記述ができない。

本研究では、大きな配列などインデックスにより識別される集合を用いる際、プロセッサ数の変化に対応した並列プログラムを簡単に記述できる、「分散集合オブジェクト」を提案し実装した。実装においては要素と断片の対応を分散保持し、メッセージの集中を防いで性能を損なわないようにした。この分散集合オブジェクトモデルの下でアプリケーションを記述し、動作の確認と性能評価を行った。

2. 提案手法

2.1 対象とする分散集合

対象とするのは、インデックスで位置を識別できる集合である。これには配列・ハッシュ表などが含まれ、頻繁に用いられる。実行時に動的にプロセッサ数が変化する場合、データとプロセッサの対応は変化するので、インデックスから位置を簡単に計算することはできない。このためインデックスと要素の対応表を保持し、必要に応じて更新する必要がある。

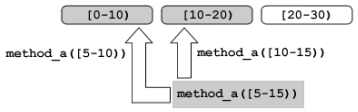

図 1 メソッド呼び出し

2.2 記述モデル

プログラマはまず断片オブジェクトを定義し、その断片が保持する範囲のデータに対する操作をメソッドとして記述する。次に、プログラマはこのオブジェクトを用いる関数を記述する。外部からオブジェクトにアクセスするには、個々の断片ではなく「全体」に対してインデックスの範囲を指定して呼び出しを行う。これにより、引数の集合と重なりを持つ断片でメソッドが呼び出される。これを図 1 に示す。

プロセッサ増減には、断片の分割・併合により対応する。プログラマはクラス定義において、断片オブジェクトの分割・併合を記述する。これにより、オブジェクトを用いる際にはシステムが提供する join や leave というメソッドを呼ぶだけで、断片のプロセッサへの割り当てや退避が実現される。

2.3 システムの実装

2.3.1 ルーティング

プロセッサ数が変化する環境においては、要素とプロセッサの対応表を動的に更新する必要がある。一プロセッサがこの表を集中管理すると実装は容易だが、このプロセッサにメッセージが集中し、性能上ボトルネックとなる。このため、本システムではルーティング表を各プロセッサが分散保持し、メッセージは各断片に直接届けられるようになっている。この実装には Phoenix ライブラリ[1] を利用している。

2.3.2 メソッド呼び出し

メソッド呼び出しはインデックス範囲を指定して行われる。呼び出しのメッセージはまず範囲の先頭のイ

† 東京大学

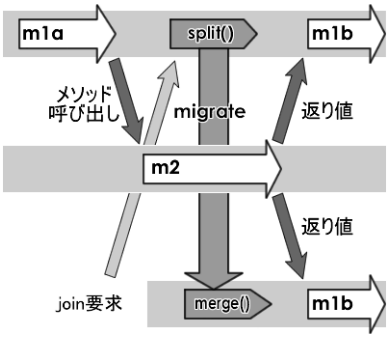

図 2 返り値を挟んでのメソッドの分割

ンデックスに送られる。受け取ったプロセッサでは、この引数を、自分が保持するインデックス集合との重なりとそれ以外の部分に分割する。そして、重なりを引数にメソッドを呼び出し、残りのインデックス集合を同様に転送する。

我々は同時にツリー状にメッセージが中継されるアルゴリズムも実装した。これは広範囲に対する呼び出しが短時間で行えるが、一回の呼び出しに対し一つの断片が複数回のメソッド呼び出しを受ける可能性がある。

2.3.3 返り値とマイグレーション

今回の実装では、システムは返り値をサポートせず、プログラマは返り値を持つメソッドを分割して記述する必要がある。この理由を以下に述べる。

返り値をサポートするシステムでは、スレッドが返り値を待ってブロックしている状況が頻繁に発生する。ここで通常のシステムではオブジェクト中のスレッドを移送できないため、マイグレーションできる機会が限定されてしまう。

そこで本システムでは、返り値を持つメソッドの呼び出しを「呼び出し前の処理」と「呼び出し後の処理」の二つに分割して記述させる。メソッド呼び出しは、呼び出し元のインデックス範囲を引数に行われ、返り値を返すメソッドは、このインデックス範囲に対し返り値に相当する値を指定して「呼び出し後」のメソッドを呼び出す。この様子を図 2 に示す。

これにより、メソッド呼び出しの各切れ目でオブジェクトのマイグレーションが可能になる。またメソッドを呼び出したオブジェクトが返り値を待っているときに分割されても、双方に返り値のデータが送信されるようなプログラムを記述することができる。

3. アプリケーションの記述例

本システムを用いて、Gauss-Seidel 法のプログラムを記述した。境界部分で隣の断片のデータを要求する際は、要求元の断片のインデックス範囲を引数に入れて、要求先のインデックス範囲に対しメソッドを呼び出す。データは要求元のインデックス範囲に対し返されるため、端の交換中であっても断片はマイグレートすることができ、プロセッサの参加・脱退に対応できる。

本プログラムはプロセッサ 81 台の環境で正しく動作し、またプロセッサ 64 台の連続 join に成功した。ここで、全体の配列の一辺を 10000, 20000, 30000、またプロセッサ数を 1-81 台の間で変化させ、一回のループにかかる時間から MFlops 値を計算した。この結果を図 3 に示す。現状ではプロセッサ数の増加に対し、頭打ちとなる傾向が得られた。原因については調査中だが、実装の改良により改善できるものと考えている。

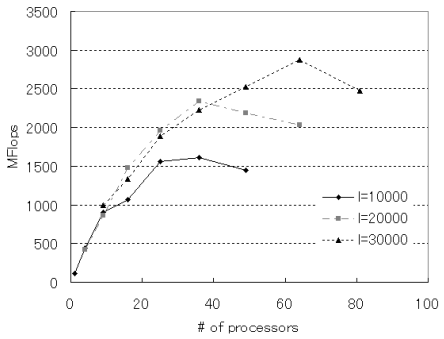

図 3 MFlops 値

4. おわりに

本研究ではプロセッサ間に分散したデータを取り扱う「分散集合オブジェク分散集合オブジェクト」を提案し実装した。

今後の課題としては、プリプロセッサによりメソッドが擬似的に返り値を取れるようにすること、実装の改良による処理系の性能向上が挙げられる。

参考文献

1) Kenjiro Taura, Toshio Endo, Kenji Kaneda, and Akinori Yonezawa. Phoenix : a Parallel Programming Model for Accommodating Dynamically Joining/Leaving Resources. ACM SIGPLAN Symposium (PPoPP 2003).
2) Andrew A. Chien, Vijay Karamcheti, John Plevyak, Xingbin Zbang . Concurrent Aggregates Language Report Version 2.0 (http://www-csag.ucsd.edu/papers/csag/external/ca-report.ps)