分散共有メモリシステムの複製管理方式

川口 昇、中村 健二、佐藤 文明、水野 忠則
静岡大学情報学部

分散システムにおけるタプルスペース (TS) 通信ではプロセス間の通信は非同期に行うことが可能である。しかし、TS サーバが何らかの障害でダウンしてしまうと、プロセス間の通信はそれ以上続行できなくなる。そこで解決策の1つとしてネットワーク上の計算機に TS の複製を配置することが考えられる。本研究ではネットワーク上の計算機に TS の複製を配置したモデルを考え、TS 通信の特性を生かしたより効率的な複製管理方式を提案し、シミュレーションを用いて本方式が従来の方式よりも応答時間の点で優れていることを示す。また、この方式は現在 UNIX システム上で実装中である。そこで本論文では、実装の方針と現在の進行状況についても報告する。

A Method of Replication Contorol on Distributed Shared Memory System

Noboru Kawaguchi, Kenzi Nakamura, Fumiaki Sato, Tadanori Mizuno
Faculty of Information, Shizuoka University

On the Tuple Space communication of the Distributed System, it is possible for processes to communicate ansynchronously each other. But if the Tuple Space server crashes for some reason, the processes can't continue to communicate any more. So we propose as one of the solution that the computers connected to the network have the replicas of the Tuple Space. In this research we deal with the model in which the computers connected to the network have the replicas of the Tuple Space, and we propose the method of the replication control using the characteristic of Tuple Space communication. And we found that this method is superior to the past by using simulation. Now we are establishing this method on UNIX system. So we report the scheme of this and the current progress.

1 はじめに

複数の計算機をネットワークで相互接続した分散システムでは、プロセス間の通信を非同期に行ないたい場合がある。これは、Linda [1] [4] というプログラミングモデルを利用することで実現できる。Linda は、タプルスペース（以下 TS と略記）と呼ばれる仮想化された共有メモリ空間と、TS に対してデータのやりとりを行うためのプリミティブを用意する。Linda は考え方が簡単明瞭で並列プログラミングの方法の一つと考えられている。

また分散システムでは、信頼性や可用性を高めたり、アクセス時間を短縮するためにファイルなどの資源に複製を持たせる場

合が多い。この場合、複製間でデータの一貫性を保つために複製管理 [2] が必要になる。

本研究では TS 通信で TS に複製を配置したモデルを考え、TS 通信の特性を生かしたより効率的な複製管理方式を提案する。

2 分散共有メモリシステム

2.1 TS 通信

Linda は、タプルスペース（以下 TS と略記）と呼ばれる仮想化された共有メモリ空間と、TS に対してデータ（タプルと呼ぶ）のやりとりを行うための 4 種のプリミティブをユーザプログラマに用意する。プログラマは、この 4 種のプリミティブを使って並列実行の単位を生成、それらの間の同期、通信を陽に記述し、プログラムを並列に実行しようとするものである。

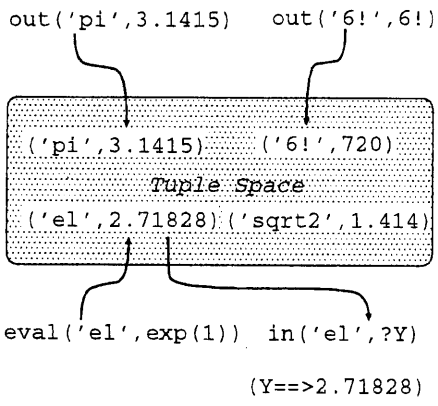

図 1: Linda モデル

TS 通信では基本的に

- **out()** : タプルを TS に置く
  (引数はその場で評価される)
- **eval()** : タプルを TS に置く
  (引数はその場で評価されない)
- **in()** : タプルを TS から取り出す
- **rd()** : タプルを TS から読み込む
  (タプルは TS に残る)

の 4 つの基本命令を用いている。このため TS 通信は以下のような特性を持つと予想される。

1. 書き込み頻度が読み込み頻度より大きい
2. タプルはファイルのように更新されることが無い

2.2 対象モデル

図 2 は複数の計算機をネットワークで接続したモデルを表している。各計算機には複数のプロセスが存在し、各プロセスは TS と呼ばれる共有メモリ空間を利用してデータの受渡しをしている。このようなモデルでは、プロセス間の通信は非同期に行える。

本研究では、図 2 を拡張して、TS の複製を各計算機に配置した図 3 のモデルを対象とする。

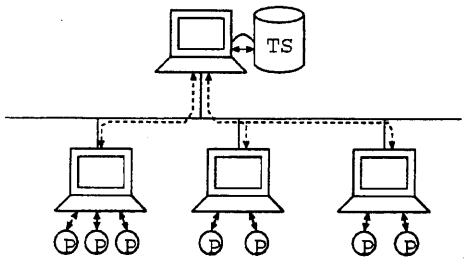

図 2: TS 通信を用いた共有メモリシステム

2.3 複製管理

図 3 のように複製を配置することにより、複製間の一貫性を維持するための複製管理

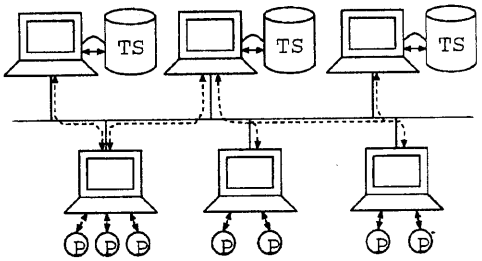

図 3: 複製付き TS 通信を用いた共有メモリシステム

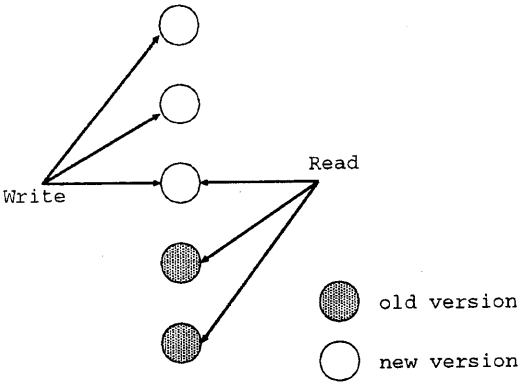

図 4: 定数合意方式

が必要になる。

複製管理には様々な方法が提案されているが、代表的な方法として、以下の 2 つを挙げる。

- 任意読み出し全書き込み方式

  データの更新は全複製に対して行われ、読み込みはその中の 1 つに対して行われる。

- 定数合意方式 (図 4)

  この方法は書き込み定数、読み込み定数を設け、それぞれの定数に応じた数だけ書き込み、読み込みを行う。ここで書き込み定数と読み込み定数の和が複製数より多くなるように設定すると、読み込みと書き込みが共に行なわれる複製が少なくとも 1 つは含まれることになり、読み込み時に最新のデータ、つまり最も最後に書き込まれたデータが得られる。データが最新かどうかは複製に付加されたバージョン番号で判断する。

定数合意方式の利点は読み込み数、書き込み数を自由に設定できる点である。例えば読み込み頻度が多ければ読み込み定数を小さくし、書き込み頻度が多ければ書き込み定数を小さくすればより少ないメッセージ数でデータのアクセスができる。

このような点から本研究では複製管理に定数合意方式を用いる。

2.4 ロック

図 2 、3 からもわかるように、TS のような共有メモリにアクセスするプロセスは複数存在する。このような場合、あるプロセスが TS にアクセス中の時は他のプロセスがこれにアクセスできないように TS にロックを掛ける必要がある。

特に図 3 のように TS の複製を配置した場合は、複数の複製にロックを掛けなければならない。この時、たとえある複製における処理が早く終わっても、次のプロセスは選択された全ての複製の処理が終わるまで待つ必要がある。つまり、複数の複製間で同期をとる必要があり、処理効率の面からは望ましくない。そこで本研究ではこれを非同期に行う方法を提案する。

3 本研究の複製管理方式

3.1 定数合意方式の適用

本研究では 2.3 の複製管理はタプル単位で行うものとする。つまり、TS そのものに対して複製を配置するのではなく、タプル１つ１つに対して複製を配置する。

2.1より TS 通信では書き込み頻度が読み込み頻度より大きいことから、2.3 でも述べたように複製管理は書き込み数を減少できる定数合意方式を用いる。したがって、平均の応答時間を短くするには書き込み数を１、読み込み数を全複製数にするのが最も効率的である。しかし、この設定では複製数が１、つまり複製が存在しなくなり、複製を配置する利点が失われる。そこで今回は、書き込み定数と読み込み定数を

（書き込み定数）＝（読み込み定数）＝
（複製数）／２＋１

とし、任意読み出し全書き込みとの中間の値をとることにした。

3.2 変更点

2.1ではタプルは更新されないことを述べた。これにより定数合意方式で用いられていたバージョン番号は不要になるため、本方式ではバージョン番号は用いない。

また、TS 通信では in() を実現するために読み込み操作の他に削除操作が必要になる。定数合意方式の条件を満たすには、削除操作は書き込み操作の時と同一のタプルの組に対して行われなければならない。そこで本方式ではタプルの書き込み操作の時に、タプルに「自分の複製が置かれた全サーバの識別子のリスト」を付加することでこれを解決する。

3.3 非同期方式

2.4 では、TS にアクセスする時は全複製にロックを掛けるため全ての複製に対する処理が終わるまで次のプロセスは処理を開始できないことを述べた。

そこで本研究ではこの待ち時間のロスをなくすため、ロックを掛けずに処理を行なう事にする。これにより、各々のプロセスから出されたメッセージを非同期に受け付けることができるようになる。そこで、以後この方式を「非同期方式」、従来のロックを掛ける方式を「同期方式」と呼ぶことにする。

非同期方式では各プロセスから出されたメッセージは図 5に示すように各複製の待ち行列に入れられる。各複製は待ち行列の先頭からからメッセージを取りだし、順に処理を行なう。ここで、各複製は他の複製の処理が終了するのを待たずに次の処理に移れるため待ち時間のロスがなくなり効率的に処理ができる。

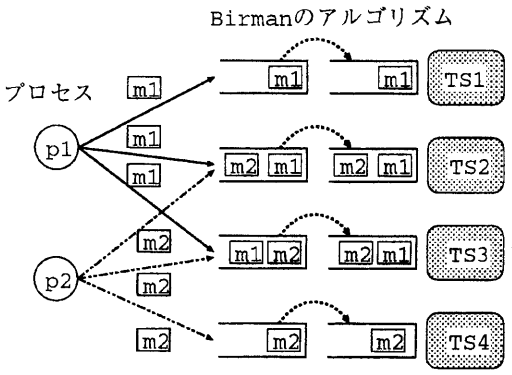

図 5: 非同期方式

3.4 Birman のアルゴリズムの適用

非同期方式は処理時間を短縮できるという点で優れているが、ロックを掛けないた

め、図5のようにメッセージの到着順序が複製ごとに異なる場合は一貫性の上で問題が生じる。そこで、本研究では待ち行列中のメッセージをBirmanのアルゴリズム[3]を用いて到着するメッセージの順序が全ての複製で同一、つまり全順序サービスになるように並び換えをしている。

従来の同期方式の応答時間は、メッセージが出されてから対象とする全ての複製の処理が終了する時刻までだった。しかし、非同期方式に全順序サービスを加えた本方式の応答時間は、メッセージが出されてから対象とする複製のうちで最も処理の早いものが終了する時刻までとなり、同期方式と比べて応答時間が短縮される。これは、あるメッセージの処理が1つでも終了すれば、他の複製でのそのメッセージの処理は将来的に行なわれることが全順序サービスで保証されるからである。例えば、図5ではメッセージm1の応答時間はメッセージが出されてからTS1～TS3で最も処理が早いものの終了時刻までになる。

4 シミュレーションによる評価

定数合意方式の同期方式と非同期方式の応答時間を比較するため、本研究ではシミュレータを作成し、評価を行った。

今回のシミュレーションでは通信時間は処理時間に比べて極めて短いものとし、通信時間を0とした。また、通信時間を0に近似することで、通信網の競合は無視してよいものとした。

シミュレーションの測定条件と結果を以下に示す。

測定条件

- プロセス数：3
- ノード数(TSの数)：4
- 書き込み定数：3
- 読み込み定数：3
- 要求の処理時間：パラメータ0.5の指数分布

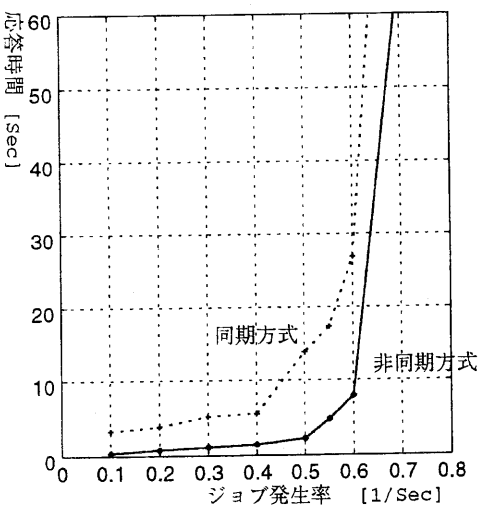

図6: シミュレーション結果

グラフからも分かるように同じ要求発生率でも非同期方式のほうがより短い応答時間で処理を行なえることがわかる。

今回のシミュレーションの対象は定数合意方式の同期方式と非同期方式だったが、これに任意読みだし全書き込み方式の同期方式と非同期方式を加える場合は、通信時間も考慮に入れて評価をしなくてはならない。

5 複製付きTS通信の実装

本研究の複製管理方式は現在UNIXのシステム上で実装中である。ここでは実装の方針と現在の進行状況について報告する。

5.1 実装の方針

システム構成は次の図7に示すとおりである。

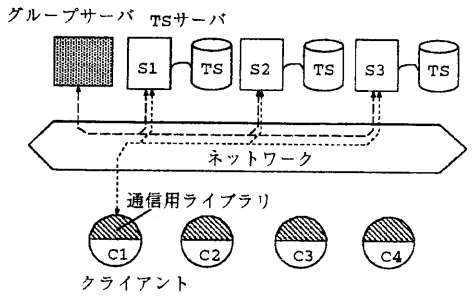

図 7: システム構成

ネットワーク上には TS を管理するための TS サーバが存在する。TS には複製を持たせるため複製の数だけ TS サーバが必要となる。TS を利用するプロセスをクライアントとすると、各クライアントは通信用のライブラリを用いて TS と通信を行う。複製管理は通信用ライブラリの中に含めるものとし、クライアントは複製に関しては意識しなくてもよい。クライアントと TS サーバとの通信はグループ通信を用いる。グループは同一の複製を持つ TS サーバから構成され、これを管理するためにグループサーバを設ける。

TS サーバとグループサーバはデーモンプロセスで動く。また、Birman のアルゴリズムは2フェーズコミットメントに基づいているため、複製管理のアルゴリズムは TS サーバと通信用ライブラリそれぞれに分割して実行される。

5.2 現在の進行状況

現時点では、まだ設計段階である。
現在、以下の点を検討中である。

- TS サーバのグループへの登録処理、離脱処理
- 処理の途中でグループ情報が変更された時の他のクライアントへの通知方法

6 まとめ

本研究では分散システムにおける複製付き TS 通信を対象とし、TS 通信の特性を生かした複製管理方法を提案した。ポイントは次の2点である。

第1に複製管理に定数合意方式を用いて TS 通信の特性に合わせて改良を加えたこと、第2に応答時間を短縮するため処理を非同期にし、一貫性を維持するため Birman のアルゴリズムを組み合わせたことである。

またシミュレーションを用いて、本方式が従来の同期方式よりも応答時間の面で優れている事を示した。

今後は本方式の実装を進めると供に、複製の配置方法についても考察し改良を加えていきたい。

参考文献

[1] 木村 康則, 住元 真司, 細井 聡, 小沢 年弘, 服部 彰 : Linda 処理系の試作について

[2] Andrew S. Tanenbaum 著, 引地 信之, 引地 美恵子 訳 : OS の基礎と応用, トッパン (1995.11)

[3] 滝沢 誠, 中村 章人 : 放送通信アルゴリズム, 情報処理, Vol.34, No.11, (1993.11)

[4] 吉田 紀彦, 樽崎 修二 : 場と一体化したプロセスの概念に基づく並列協調処理モデル Cellula, 情報処理学会論文誌, vol.31, No.7 (1990.7)

[5] K. Birman, A. Schiper, P. Stephenson : Lightweight Causal and Atomic Group Multicast, ACM Transactions on Computer Systems, vol.9, No.3 (1991.8)