マルチメディア通信と分散処理 82－22
（1997. 4. 25）

複製付きタプルスペース通信の管理方法と実装

川口 昇、佐藤 文明、水野 忠則
静岡大学情報学部

分散システムでは共有メモリを利用して複数のプロセスを並列に動作させることが可能である。しかし、共有メモリを管理するサーバに障害が生じるとそれを利用している全プロセスが停止してしまう問題がある。これを解決する方法の１つとして、ネットワーク上に共有メモリの複製を配置する方法が考えられる。本研究では共有メモリとしてLindaモデルのタプルスペース通信を対象とし、タプルスペース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。また、この方式は現在UNIXシステム上に実装中である。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 Control and Implementation of Replicated Tuple Space Communication

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

On the Distributed System, it is possible for processes to run in parallel by using shared memory. But if the server controling the shared memory crashed, all the processes using it will stop. As one of the solution of this we made the computers connected to the network had the replicas of the shared memory. In this research we deal with the Tuple Space Communication of Linda as the shared memory, and we propose the method of the replication control using the characteristic of Tuple Space communication. Now we are implementing this method on UNIX system. So we will report the scheme of this and the current progress.

1 はじめに

分散システムでは共有メモリを利用して複数のプロセスを並列に動作させることが可能である。共有メモリのモデルとしてはLinda [1]、Cellula [2] など様々なモデルが提案されている。特にLindaは場を媒体とした通信を行うモデルで、並列プログラム[1] を書くときによく用いられる。

ところがこのような共有メモリを利用して通信する場合、共有メモリを管理するサーバに障害が生じると、それを利用している全プロセスに影響が及ぶという問題がある。これを解決する方法の１つとして、ネットワーク上に共有メモリの複製を配置する方法が考えられる。

本研究ではLindaモデルのタプルスペース通信でタプルスペースの複製を配置したモデルを考え、この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、その実装方法について報告する。

2 分散共有メモリシステム

2.1 TS 通信

本研究では分散共有メモリとして Linda モデルのタプルスペース通信を扱う。Linda モデルは図 1のように、タプルスペース（以下 TS と略記）と呼ばれる共有メモリ空間と、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 種の基本命令を提供する。基本命令は out()、eval()（タプルを TS に置く)、in()（タプルを TS から取り出す)、rd()（タプルを TS から読み込む）の 4 つで、プログラマは、この基本命令を使って並列実行の単位を生成し、プログラムを並列に実行できる。

このような TS 通信は、書き込み頻度が読み込み頻度より高い、タプルはファイルのように更新されることが無いという 2 つの特性を持つ。

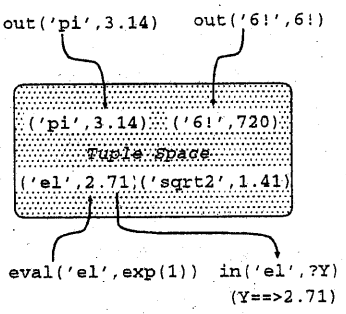

図 1: Linda モデル

2.2 複製付き TS 通信

従来の TS 通信では共有メモリである TS に障害が生じた場合にその影響が全クライアントに及ぶという問題があった。そこで本研究では TS の複製をネットワーク上に配置した図 2 に示されるモデルを対象とする。このようなモデルを利用した TS 通信を以後「複製付き TS 通信」と呼ぶ。

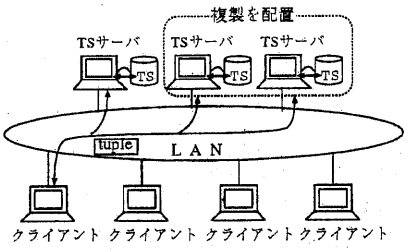

図 2: 複製付き TS 通信を用いた共有メモリシステム

3 提案する複製管理方式

3.1 定数合意方式

2.1節で TS 通信では書き込み頻度が読み込み頻度よりも高いことを述べたが、本研究ではこの特性を生かして複製管理に定数合意方式 [4] (図 3) を採用する。この方式は書き込み定数 W、読み込み定数 R をそれぞれ設け、各定数の数だけ読み込み及び書き込みを行う方式である。この時、W と R の和が複製数 N より大きくなるように設定すると、読み込み、書き込みが共に行われるものが少なくとも 1 つは存在する。従って、読み込み時に最新のデータが得られ、一貫性が保証される。このときデータが最新かどうかは複製ごと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頻度、書き込み頻度に合わせて各定数を設定できる点で、書き込み頻度が高い TS 通信では書き込み定数を小さくすることで全体の処理時間を短縮できる。また本来この方式はファイルの複製に対して用いられる方法であるため、これを TS 通信に適用するには幾つかの変更が必要になる。

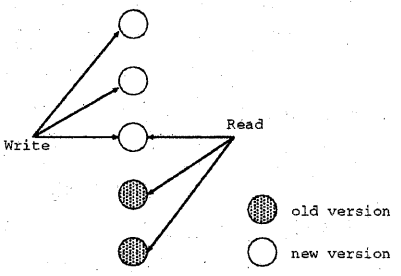

図 3: 定数合意方式

3.2 改良点

3.2.1 バージョン番号

2.1節で、タプルは更新されることがないため、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。また、TS 通信では in() を実現するため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ときと同一のタプルの組に対してなされる必要がある。そこで 書き込み操作のときに、「他の複製の位置を示す識別子」をタプルに付加することでこれを解決する。

3.2.2 非同期処理

分散システムで共有領域にアクセスするときは他からのアクセスがないように共有領域にロックを掛ける必要がある。しかし今回のように共有領域が複数ある場合でも、通常は全複製にロックを掛ける必要があり、全ロックが解除されるまでの待ち時間が無駄になる。そこで本研究では複製にロックを掛けずに非同期に処理することにする。

非同期方式は処理時間の短縮という面では優れているが、ロックを掛けないためメッセージの到着順序が複製ごとに異なる場合があり、一貫性が保証されなくなる。そこで、図 4 のように待ち行列中のメッセージを Birman のアルゴリズム [3] を用いて並び換え、到着するメッセージ順序が全ての複製で同一になるようにしてこれを解決する。

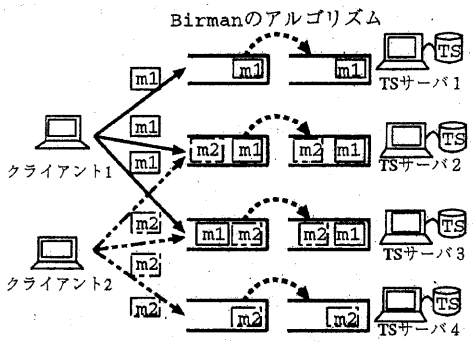

図 4: 非同期方式

4 シミュレーションによる評価

従来の同期方式と本方式の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。図 5はプロセス数を 3、TS の数を 4、要求の処理時間をパラメータ 0.5 の指数分布とした時の応答時間をグラフにしたものである。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を得られることがわかる。

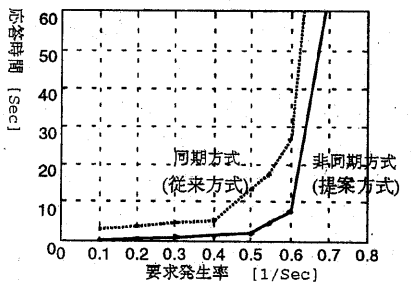

図 5: シミュレーション結果

5 複製付き TS 通信の実装

複製付き TS 通信は現在 UNIX のシステム上に実装中である。この章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 システム構成

システム構成は次の図 6 のようになる。

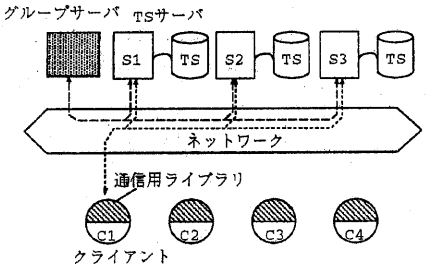

図 6: システム構成

このシステムは TS サーバ、クライアントプロセス、グループサーバからなる。

TS サーバは TS で送受信されるタプルの管理を行なうサーバで、タプルの管理の他に後述のクライアントプロセスから出された要求をキューに貯める働きもする。本システムは TS を複数配置したモデルを対象とするため、TS の数だけ TS サーバを用意する。TS サーバは「要求キュー」、「実行キュー」の 2 つのキューを持っている。要求キューはクライアントからの要求が入るキューである。システムはこのキューの各要求に付随するタイムスタンプを付け換え、実行可能状態にする。実行可能状態になった要求は実行キューに入れられタイムスタンプ順にソートされる。またこの処理とは並行に、システムは実行キューの先頭から要求を実行していく。

クライアントプロセスは実際に TS 通信を利用するユーザによって生成されるプロセスである。ユーザは複製の存在を気にせずに TS 通信を行なえるように、複製管理アルゴリズムは通信用ライブラリに含める。

提案する複製管理アルゴリズムはクライアントと TS サーバ間のメッセージ交換で実現されるため、TS サーバにも複製管理アルゴリズムを実装する必要がある。このときクライアントプロセスは複数の TS サーバと通信する必要がある。そこでこれらの通信を容易にするため、クライアント-TS サーバ間の通信はグループ通信で実現する。すなわち、複数の TS サーバで 1 つのグループを構成し、クライアントは TS サーバ個々ではなく、このグループに対して通信を行うようにする。またグループ通信を用いることにより、障害時の回復処理がグループの離脱処理、登録処理という形で比較的容易に実現できる。本システムではグループ通信を実現するために、グループのメンバ情報 (以後「グループ情報」と呼ぶ) 等を管理するグループサーバを新たに配置する。

グループ情報にはグループのメンバ数、各メンバのアドレス等が含まれ、クライアントはこの情報をもとにグループ通信を行う。通常グループ情報はグループサーバと各 TS サーバがそれぞれ保持する。グループの状態が変化した場合は、まずグループサーバのグループ情報が変更され、そのあと各 TS サーバのグループ情報が変更される。このようにしてグループサーバ、TS サーバ上には常に最新のグループ情報が置かれている。

5.2 システムの動作

この節ではシステムの動作について説明する。ここではグループを構成するためのグループ登録処理について述べ、タプル操作命令の実行を out() 命令を例にとって説明する。また、サーバに障害が生じた時のグループ離脱処理についても説明する。

5.2.1 グループ登録処理

まずグループを構成するにはTSサーバが起動した時に図7に示すようなグループ登録処理が必要になる。

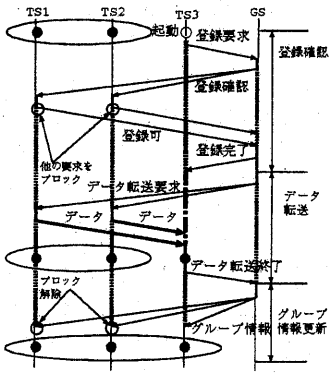

図 7: グループ登録処理

グループ登録処理は、登録確認処理、データ転送処理、グループ情報更新処理の3つから成っている。

最初の登録確認処理は新しいメンバをグループに登録してもよいかを既存のグループのメンバに確認する処理である。登録が可能な場合は、既存のメンバは現在処理中の処理を完了し、以後の要求をブロックする。

次のデータ転送処理は既存のメンバのTSデータを新メンバのTSにコピーする処理である。このとき複製間の一貫性を維持するためにデータ転送は読み込み定数の数だけ行う必要がある。

最後のグループ情報更新処理は新メンバをグループに加えた新たなグループ情報をグループのメンバに送信する処理である。また、この処理が終了すると同時に要求のブロックは解除される。

5.2.2 命令の実行

次に実際のタプル操作命令の実行を、図8に示される out() 命令を例にとって説明する。

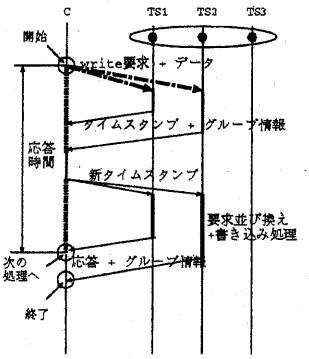

図 8: out() 命令の実行

クライアントで out() 命令が出されると、クライアントは write 要求と対象となるデータをグループのメンバの一部に送信する。送信すべきメンバの数は書き込み定数に従ってランダムに選択される。write 要求を受信したメンバは各自が保持する要求キューにこの要求を入れ、各メンバにローカルに用意されたタイムスタンプ (シーケンス番号) を送り返し、自分のタイムスタンプをインクリメントする。クライアントは要求を出した全メンバからタイムスタンプを受信すると、その中の最大値を選択し、これを再びグループのメンバに送信する。これを受信したメンバは、要求キューに貯められた要求に付随するタイムスタンプを、この新しいタイムスタンプに付け換えて実行キューに入れる。各メンバはこれらの処理と並行に、実行キューに入れられた要求を先頭から順に実行し、クライアントに応答を返す。このときクライアントは全メンバからの応答を待つ必要はなく、最初の応答が来た時点で次の処理に移ることができる。これは

少なくとも1箇所で処理が完了すれば、残りの処理が失敗しても、後述するように失敗したサーバをグループのメンバから外すことで一貫性が保証できるからである。

5.2.3 グループ離脱処理

次にTSサーバがダウンして要求に対する処理が失敗した場合について説明する。サーバのダウンは応答時間にタイムアウトを設けることで検出される。

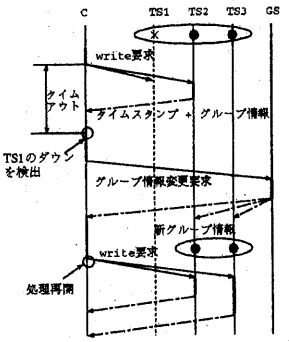

図 9: グループ離脱処理

例えば図9のようにTS1がダウンした場合は最初の応答でタイムアウトになり、TS1のダウンが検出される。そしてクライアントはTS1のダウンをグループサーバに通知する。グループサーバはこれを受信して、TS1をグループから外した新たなグループ情報をグループの他のメンバとクライアントに送る。そしてクライアントはこの新しいグループ情報を元に先程の処理を再開できる。もしその後TS1が回復した場合は、再びTS1の登録処理を行えば良い。

5.3 現在の進行状況

現在グループサーバ、TSサーバ、通信用ライブラリの通信部分が完成しており、通信部分の性能評価を行っている段階である。しかし現段階ではTSの実装がまだ終わっていないため、今後はそれらの実装を完了し、完成したシステムで性能評価を行う予定である。

6 まとめ

本研究では従来のTS通信を拡張した複製付き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。提案方式ではまず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。そして応答時間を短縮するため処理を非同期にし、一貫性を保証するためBirmanのアルゴリズムを組み合わせた。

また提案方式を実装するときの実現方法と現在の進行状況についても述べた。

今後は実装を完成させると共に複製の配置方法についても考察し改良を加えていく。

参考文献

[1] Nicholas Carriero, David Gelernter : How to Write Parallel Programs : A guide to the Perplexed, ACM Computing Surveys, Vol.21, No.3, (1989)

[2] 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)

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

[4] D.K.Gifford: Weighted voting for replicated data, in Proc. 7th ACM SIGOPS Symp. Oper. Syst. Princip. (1979)