2F3-01

グリッド環境のためのロバストなデータ共有システム

Robust Data Sharing System for Grid Environment

嶺 行伸[*1] 平石 広典[*2] 溝口 文雄[*3]
Yukinobu Mine Hironori Hiraishi Fumio Mizoguchi

[*1]東京理科大学理工学部
Faculty of Science and Technology, Tokyo University of Science

In the grid environment which calculates by a lot of cheap computers, a method is important that storing and reference the data shared between the threads performed by different computer. In this paper, we propose the algorithm and the protocol for recording data distributive and keeping the contents same. By distributing data, the danger can be avoided that lose the data by a sudden trouble. It is suitable for the grid environment that uses a lot of computers.

1. はじめに

近年、パーソナルコンピュータの飛躍的な性能向上により、安価にギガフロップス単位の計算能力、ギガバイト単位のメモリ、数百ギガバイト単位の補助記憶装置を手に入れる事が可能となった。

しかし、一般的な事務処理では、これらの能力は必要とされない。計算能力、メモリ、補助記憶といったリソースの大半が遊休状態にあるが、補助記憶の一部をネットワークを介して利用できる程度であり、各端末のリソースが個別に遊休状態である事には変わりがない。

複数の計算機のリソースを統合するシステムとしては、計算能力を統合するグリッド、メモリを統合する分散共有メモリ等がある [1][2]。これらのシステムは、複数の計算機を利用するため、計算機の台数が増えるに従って故障率も増大する。そのため、システムを構成する計算機に障害が発生しても、システム全体が停止しないロバスト性が求められる。

ロバストなデータ管理の方法としては、補助記憶装置を対象とした RAID システムの手法を参考にする事ができる。分散共有メモリに RAID と同様の冗長性を持たせる事によって、複数の計算機で構成される分散共有メモリシステムの耐障害性を向上させる事ができる。

2. 目的

既設置の計算機の遊休資源を利用するシステムは、特定のサーバーに管理させるのではなく、各計算機が自律的に他の計算機にアクセスするアドホックなシステムである事が望ましい。なぜなら、サーバーで集中管理する場合、管理する計算機の数に比例して大規模なサーバーシステムが必要となるからである。

本研究では、アドホックに構成されるグリッド環境を実現するためのデータ共有システムを設計し、そのシステムにロバスト性を持たせるためのデータの分散格納方式、通信プロトコルを提案する。

3. 設計方針

我々は、アドホックなデータ共有を実現するために、以下のような設計方針をとった。

連絡先: 千葉県野田市山崎 2641 東京理科大学情報メディアセンター mine@imc.tus.ac.jp

マルチプラットフォーム 既設置の計算機の遊休資源を利用するため、Windows、Mac、Unix 等、様々なプラットフォームで利用できる必要がある

マルチキャスト通信 複数の計算機に同報するための手段として最も適しているマルチキャストを用いる

対称的なプロトコル 特定のサーバーを介した通信ではなく、各計算機が自律的に通信するため、通信元と通信先が同一のプロトコルを対称的に用いる

4. Java による共有空間のモデリング

本研究ではマルチプラットフォーム性を実現するために、実装言語としては Java を用いた。共有変数の読み出し・書き込みと同期処理を行う SharedObject クラス、1 つの計算機のスコープを管理する Computer クラス、及び複数の計算機にまたがったスコープを実現する VirtualComputer クラスを用いて、共有空間のモデリングを行った。

4.1 SharedObject クラス

このクラスは、共有変数をラップするクラスであり、変数の読み出し、更新、ロック、アンロックを実装する。

getObject() 共有変数の値を読み出す

update(Object) 他の計算機に更新通知を送る

lock() 変数をロックする

unlock() 変数をアンロックする

4.2 Computer クラス

このクラスは、1 つの計算機に存在する共有変数スコープを管理する。アプリケーションは、このクラスを通して共有変数へのアクセスを行う。

getSharedObject(String) 変数をスコープから取得する

share(SharedObject) 変数をスコープに登録する

4.3 VirtualComputer クラス

VirtualComputer は Computer のサブクラスであり、共有空間を構成する端末全てが登録される。各端末は、VirtualComputer オブジェクトを通して互いの存在を知る。

VirtualComputer は、Computer の提供する全てのメソッドを拡張し、ローカルで発見できなかった共有変数について、マルチキャストを用いたプロトコルで問い合わせを行う。

5. データ共有プロトコル

本研究では、マルチキャストを用いた対称型データ共有プロトコルを用いる。このプロトコルは、データの問い合わせ、更新、同期の機能を提供し、それぞれの送受信を各計算機が等しく担当する事によって対称性を実現する。

マルチキャストを用いる事には、以下のような利点がある。

- 共有空間に参加する計算機の数に通信量が比例しない
- 特定の計算機の障害によって通信が阻害されない
- 複数の計算機による冗長なデータ記録を簡単に実現できる

5.1 データの問い合わせ

本システムにおいて、共有空間に対してデータの問い合わせを行う時は、以下のような点を考慮する必要がある。

- データは複数の計算機で保持されている
- ある時点において、各計算機で保持されているデータが同一である保証はない
- 問い合わせによってデータを取得した計算機は、そのデータを保持する計算機の１つとなる

複数の計算機で保持されている、必ずしも一致しないデータから最新の情報を選び出すために、データにタイムスタンプを導入し、複数の計算機にタイムスタンプの問い合わせを行う仕組みを導入した。そのプロトコルを、データ要求側と提供側に分けて説明する。

データ要求側の動作は、データ検索要求とデータ要求の２つに分けられる。

- マルチキャストグループに対して変数名、タイムアウト期限を指定してデータ検索パケットをマルチキャストする
- タイムアウト時間まで検索応答パケットを待つ
- 複数の検索応答パケットから、最新のタイムスタンプのデータを持つ計算機を特定し、データ要求パケットをユニキャストする
- データ提供パケットを受信したら、変数名をスコープに追加する
- データを展開する

データ提供側も、検索応答とデータ提供に分けられる。

- データ検索パケットを受信したら、タイムアウト期限を過ぎていれば破棄する
- スコープから変数名を検索し、存在すればタイムスタンプを含む検索応答パケットをユニキャストする
- データ要求パケットを受信したら、スコープから該当する変数名を探し、なければ破棄する
- データ提供パケットをユニキャストする

5.2 データの更新

データの更新はマルチキャストによって行い、空間に参加する全ての計算機が更新データを受信する。更新データはキャッシュに蓄えられ、キャッシュに存在するデータはスコープにあるのと同じ扱いになる。キャッシュの容量が制限に達した場合、ランダムに選んで破棄する。

- スコープ内で保持しているデータを更新する
- 変数名、タイムスタンプ、データを含む更新通知パケットをマルチキャストする

更新受理側は、受理側がデータを保持している場合と、保持していない場合によって動作が異なる。

- 更新告知パケットを受信したら、スコープ内で保持している変数であれば更新する
- スコープに存在しない場合はキャッシュに保存する

5.3 同期処理

同期処理もマルチキャストによって行う。ロックとアンロックによってクリティカルリージョンを設定し、あるデータを同時にロックできるのは１つの計算機だけである。

ある変数をロックする時、既にロックされていれば処理はブロックする。ロックされていなければ、変数名を指定してロック通知パケットをマルチキャストし、変数をロックする。ロック通知パケットを受信した側は、指定された変数がスコープにあった場合、その変数をロックする。

ある変数をアンロックする時は、変数名を指定してアンロック通知パケットをマルチキャストする。アンロック通知パケットを受信した側は、指定された変数がスコープにあった場合、その変数をアンロックする。

6. まとめ

本論文では、遊休資源となっている計算機のメモリを統合して１つの大きな共有空間を構成し、その上でアドホックなデータ共有を行うためのプロトコルついて述べた。各計算機は互いに対称的なマルチキャスト通信を行うことで、特定のサーバーに依存しない柔軟な構成を可能にしている。

また、マルチキャストによるデータの更新によって、複数の計算機のキャッシュにデータの複製を持たせ、システムを構成する計算機の一部に障害が発生した時にデータを喪失する危険性を回避できるロバスト性を実現した。

参考文献

[1] Yu, W. and Cox, A.: Java/DSM: a Platform for Heterogeneous Computing, ACM 1997 Workshop on Java for Science and Engineering Computation, Vol. 43.2, pp. 65-78(1997)

[2] Aridor, Y., Factor, M. and Teperman, A.: cJVM: a Cluster Aware JVM, Proceedings of International Conference on Parallel Processing ’99, pp.31-39(1999)

[3] 嶺 行伸, 西山 裕之, 溝口 文雄, スマートオフィスにおけるマルチメディア協調を実現するコンポーネント間協調プロトコル, 人工知能学会第 16 回全国大会 (2002)