5ZA-08　分散共有メモリシステム *Lemuria* における一貫性制御プロトコル選択機構の構築

栢 武司[†]　近藤 照之[†]　斎藤 彰一[††]　大久保 英嗣[†††]
[†]立命館大学大学院理工学研究科　[††]和歌山大学システム工学部情報通信システム学科
[†††]立命館大学理工学部情報学科

1　はじめに

我々が開発を進めている *Lemuria*[1] は, 分散共有メモリを操作する機能を提供する分散並列処理のためのプラットフォームであり, PC や WS といった一般的な計算機をその対象としている.

Lemuria では, 階層的な計算機クラスタ構成を用いて分散共有メモリを実現している. この階層的な構成によって, 分散共有メモリ機能を実現するために必要となる通信量を抑制することができ, 計算機の数が数百台規模の環境においても分散共有メモリシステムの構築を可能としている.

現在, *Lemuria* において採用されている一貫性制御方式は 1 つの方式のみである. そこで, *Lemuria* において複数の一貫性制御方式を採用し, アプリケーションの種類に応じて適切なプロトコルを選択可能とすることを考えている. これにより, アプリケーションの高速化や特定のアプリケーションに特化したプロトコルの実装が可能になると考えている.

2　*Lemuria* の構成と特徴

本章では, *Lemuria* の構成と特徴について述べる.

2.1　*Lemuria* の構成

Lemuria は, サーバである Reflector と Lemuriad 及びユーザに分散共有メモリ機能を提供するライブラリである libLemuria で構成される.

- Reflector
  複数の Lemuriad を管理する上位のサーバプロセスである.
- Lemuriad
  クライアントクラスタごとに 1 つ配置される *Lemuria* のサーバプロセスである.
- libLemuria
  Lemuria の機能を提供するライブラリである. ユーザプログラムにリンクして使用する.

また, *Lemuria* における計算機クラスタは 3 階層構成となっており, 以下の 3 種類のクラスタからなる. 図 1 に *Lemuria* の全体構成を示す.

- ルートクラスタ
  すべての Reflector によって構成される.
- サーバクラスタ
  複数の Lemuriad とそれらを管理する 1 つの Reflector によって構成される.
- クライアントクラスタ
  複数のクライアントノードとそれらを管理する 1 つの Lemuriad によって構成される.

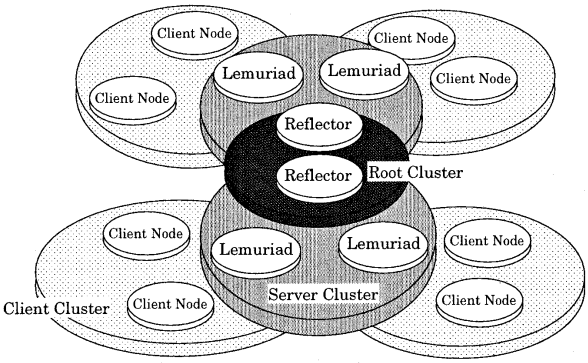

図 1: *Lemuria* の全体構成

2.2　*Lemuria* の特徴

Lemuria では, 前節で述べたような階層化された構成により, 分散共有メモリを実現している. この構成は, 1 つの計算機の接続の最大数を限定し, 計算機数の増加に起因する一貫性制御のための通信量を抑制する効果がある. また, *Lemuria* では, 上記のクラスタ構成の適用およびクラスタ単位でのメモリオブジェクトのキャッシュの適用により, 計算機の数が数百台規模の分散環境においての適用を可能にしている. 以上のことから, *Lemuria* の主要な特徴は, クラスタの概念を用いた階層的な分散環境を実現していることであると言える.

An Implementation of Multiple Consistency Protocols on Distributed Shared Memory System *Lemuria*
Takeshi Kaya[†] Teruyuki Kondo[†] Shoichi Saito[††] Eiji Okubo[†††]
[†]Graduate School of Science and Engineering, Ritsumeikan University
[††]Department of Computer and Communication Science, Faculty of Systems Engineering, Wakayama University
[†††]Department of Computer Science, Faculty of Science and Engineering, Ritsumeikan University

3 一貫性制御プロトコル選択機構

Lemuria では, 一貫性制御プロトコルとして Cluster-based Release Consistency(以下, CRC)[2] を採用している. CRC は, Release Consistency を階層化された計算機クラスタに適応するように拡張したものである. 現在 *Lemuria* では, CRC のみの実装であるが, 複数の一貫性制御プロトコルを選択可能とすることで, アプリケーションにとって適切な環境を選択し, 動作させることができると考えている.

3.1 効果

本機構を *Lemuria* に採用することによって得られる効果として, 以下のものが挙げられる.

- アプリケーションの持つ特性によって, 適切な一貫性制御プロトコルを選択することが可能となる.
- 特定のアプリケーションに特化した一貫性制御プロトコルを作成し, 利用することが可能となる.
- 複数のアプリケーションが同時に各々適切な一貫性制御プロトコルで動作可能となる.
- 使用する一貫性制御プロトコルのユーザによる選択が可能となる.

3.2 構成と処理の流れ

本節では, *Lemuria* における一貫性制御プロトコル選択機構の構成と処理の流れについて述べる. 図 2 に *Lemuria* における一貫性制御プロトコル選択機構の構成を示す. また, 以下では処理の流れについて述べる. ここでの番号は, 図 2 のものと対応している.

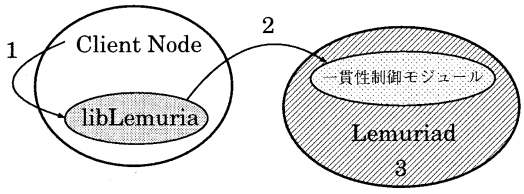

図 2: 一貫性制御プロトコル選択機構の構成

1. クライアントノードで実行されているアプリケーションプログラムにおいて, 分散共有メモリ機能を提供するためのライブラリ関数が呼び出された場合や自ノードに複製を持たない共有メモリオブジェクトのページへのアクセスが発生した場合, libLemuria に制御を移す.
2. libLemuria において要求された処理の種類及び一貫性制御プロトコルを識別するためにアプリケーションプログラムごとに設定された識別子により, 処理を要求するメッセージを作成し Lemuriad の一貫性制御モジュールに送信する.
3. 一貫性制御モジュールにおいて, 受信したメッセージの種類とメッセージに付加された識別子によって, 要求された処理内容及び一貫性制御プロトコルを判別し, 適切な一貫性プロトコルに準じた処理へと移行させる.

以上のような構成により, 一貫性制御プロトコル選択機構の構築を行う.

3.3 一貫性制御プロトコル

Lemuria では以下の一貫性制御プロトコルを階層構造に適応させた形で実装する予定である.

- Sequential Consistency(以下, SC)
  複製間の一貫性制御のための通信が多く発生するため, 速度は遅いが厳密な一貫性制御が行える. また, 共有変数へのアクセスのために, プログラム中に特別なコードを挿入する必要がないためにユーザの負担が少ない.
- Lazy Release Consistency(以下, LRC)
  共有変数の獲得アクセス時に, その時点で所有する共有変数のなかで, 他のプロセッサによってすでに更新されたものに関してのみ更新処理を行う. このため, SC と比較して通信回数を削減できる. この方式では, 排他制御及び同期制御が必要である.
- Entry Consistency(以下, EC)
  共有変数と同期変数を関連付けることにより, 各同期点においてすべての共有変数の一貫性制御を行わない. EC は, 関連付けを利用し獲得アクセス時に関連する共有変数の最新内容のみ取得する. このため, LRC と比較して通信回数を削減できるがユーザへの負担は大きい. この方式においても排他制御及び同期制御が必要である.

4 おわりに

本稿では, 分散共有メモリシステム Lemuria における一貫性制御プロトコル選択機構の構築について述べた. 現段階では本機構は実装中であり, 今後は新たなアプリケーション及び対応する一貫性制御プロトコルの追加と新たな一貫性制御プロトコルの利用に必要となるライブラリ関数の追加を行っていく予定である. また, 既存のライブラリ関数の機能拡張も必要である.

参考文献

[1] 斎藤 彰一, 國枝 義敏, 大久保 英嗣 : 分散並列処理のためのプラットフォーム *Lemuria* における分散共有メモリの性能評価, 電子情報通信学会論文誌 D-I, Vol. J82-D-I, No. 3, pp. 457-466 (1999).

[2] 斎藤 彰一, 國枝 義敏, 大久保 英嗣 : 広域分散環境における分散共有メモリの実現とその性能評価, 情報処理学会論文誌, Vol. 40, No. 6, pp. 2563-2572 (1999).