システムソフトウェアと
オペレーティング・システム 74−32
マルチメディア通信と分散処理 81−32
（1997. 2. 28）

高信頼化ミドルウェア“ARTEMIS”の設計と実装
(Advanced Reliable disTributed Environment MIddleware System)

白木原 敏雄[1] 平山 秀昭[2] 金井 達徳[1]

[1](株) 東芝 研究開発センター 情報・通信システム研究所
[2](株) 東芝 情報通信システム技術研究所

分散システム全体の高信頼化を目的に、高信頼化ミドルウェア ARTEMIS (Advanced Reliable disTributed Environment MIddleware System) を開発した。ARTEMIS はプロセス単位のチェックポイント方式を分散システムまで拡張し、分散システム全体に障害復旧機能を提供する。本稿では、ARTEMIS の概要およびそのキー技術について述べる。ARTEMIS はイントラネット上の 3 層クライアント・サーバシステムやグループウェアをターゲットにそれらの高信頼化を図る。ARTEMIS は分散チェックポイント技術、分散レプリケーション技術、ジャケットルーチン技術の 3 つのキー技術からなる。分散チェックポイント技術により、分散システム上でプロセス間通信を行なうプロセス群のチェックポイントを矛盾無く生成する。また、分散レプリケーション技術により、あるサーバ計算機の障害に対して、別のサーバ計算機に処理を途中状態から引き継ぐ。さらに、ジャケットルーチンによりオペレーティングシステムやアプリケーションの変更なしにプロセス単位のチェックポイント生成を可能にする。

Design and Implementation of ARTEMIS
(Advanced Reliable disTributed Environment MIddleware System)

Toshio SHIRAKIHARA[1] Hideaki HIRAYAMA[2] Tatsunori KANAI[1]

[1]Communication and Information Systems Research Laboratories,
Research and Development Center,
TOSHIBA Corporation
[2]Information & Communication Systems Laboratory,
TOSHIBA Corporation

For the purpose of providing higher reliability to distributed systems, we have developed “ARTEMIS” (Advanced Reliable disTributed Environment MIddleware System) which is a middleware to increase reliability of application programs. In this paper, we explain overview and key mechanisms of ARTEMIS. Distributed checkpointing mechanism creates consistent checkpoints of distributed processes. Distributed replication mechanism enables processes being taken over between two server computers. Jacket routine provides a checkpointing mechanism which creates checkpoints of processes outside the UNIX kernel and any user process can be checkpointed without rewriting source programs or re-linking binaries.

1 はじめに

近年、イントラネット上でのデータベース (DB) 処理をクライアント・サーバ (C/S) システム上で行う等、分散処理が広く使用されるようになっている。イントラネットで使用されるワークステーション (WS) やパーソナルコンピュータ (PC) は比較的信頼性が低いため、これらの計算機で構成される分散システムではさらに信頼性が低下する。イントラネットの場合、特にサーバ計算機の信頼性が重要になる。なぜなら、サーバ計算機に障害が発生した場合、それまでのユーザの処理結果が無効になり、さらにサーバ計算機上のサーバプログラムが動作するまでサービスが使用できなくなるからである。

高信頼化ミドルウェア ARTEMIS[1, 2, 3] (Advanced Reliable disTributed Environment MIddleware System) は分散システム全体の高信頼化を実現するものであり、基本メカニズムとしてチェックポイント・リスタート方式を採用している。すなわち、各プロセス単位でチェックポイント (CP) を生成する機能を持ち、さらに複数の計算機上でプロセス間通信 (IPC) を行ないながら処理を進めるプロセス群の CP を矛盾無く生成し、障害発生時には、最も最近の CP からリスタートする。ターゲットシステムは、イントラネット上の 3 層 C/S システム[1]やグループウェアであり、ARTEMIS はこれらを対象にシステム全体の高信頼性を実現する。

ARTEMIS は以下の 3 つの特徴を持つ。

- 分散システム全体の高信頼性を実現する。
- 2 台構成のサーバ計算機で、1 台に障害が発生した場合に、実行途中の状態の処理を他方のサーバ計算機上に引き継ぐ。
- ミドルウェアのみで実装しており、特別なハードウェア (HW) の追加やオペレーティングシステム (OS)、アプリケーションの変更が不要である。

これらの特徴は**分散 CP 技術**、**分散レプリケーション技術**、**ジャケットルーチン**の 3 つのキー技術によって実現している。ジャケットルーチンにより、アプリケーションや OS、HW の変更を必要とせずプロセスのチェックポイントを生成し、さらに分散 CP 技術により、IPC を行なう複数のプロセス群の CP を矛盾無く生成し、分散レプリケーション技術により、2 台のサーバ計算機間でファイルおよび CP 情報の複製をサポートすることで、サーバ計算機障害時に他のサーバ計算機で処理を引き継ぐことを可能にする。これらの技術は、マニュアル等に記載されているオープンな情報のみを使用したミドルウェアとして実現している。本稿では、ARTEMIS の概要およびこれらのキー技術について説明する

[1] DB サーバ、Web サーバー、Web ブラウザの 3 層で構成されるシステム。

2 ARTEMIS の概要

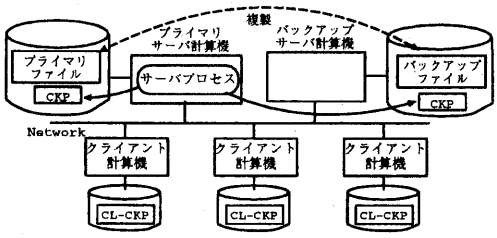

図 1: ARTEMIS のシステム概要

図 1 は ARTEMIS の対象とするシステムの構成を示したものである。2 台のサーバ計算機が存在し[2]、プライマリ・サーバ計算機上で動作するサーバプロセスとクライアント計算機上のクライアントプロセスは全体で同期して CP を生成する。さらにプライマリ・サーバ計算機上の CP はバックアップ・サーバ計算機上に転送される。また、サーバ計算機上の DB ファイル等ローカルファイルがバックアップ・サーバ計算機上でも使用できるようにする為に、バックアップ・サーバ計算機上に複製ファイルをもつ。複製ファイルの内容はチェックポイント生成のタイミングに合わせて同期する。このように定期的にチェックポイント生成を行ないながら処理を進めている途中で、プライマリ・サーバ計算機に障害が発生した場合、クライアント計算機上のプロセスはローカルディスクに保存したチェックポイント情報よりリスタートする。プライマリ・サーバ計算機上で動作していたプロセスは、バックアップ側にも CP 情報と使用していたファイルがあるため、バックアップ・サーバ計算機上でリスタートで

[2] イントラネットでは、通常複数のサーバ計算機が存在するため、あるサーバ計算機の障害発生時に他のサーバ計算機に処理を引き継ぐことができ、新たな計算機を必要とはしない。

きる。なお、ARTEMISではクライアント計算機もしくはクライアントプロセス障害の場合、リスタート処理を行なわず、そのまま、異常終了させる。なぜなら、3層C/Sシステムにおいて、1つのクライアント計算機障害で、サーバおよび他のユーザのクライアントプロセスがリスタートするのは現実的でないからである。

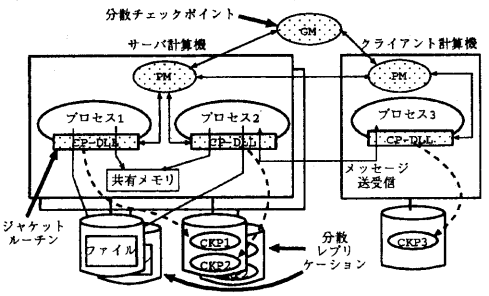

図2: ARTEMISのシステム構成

図2はARTEMISのシステム構成を示したものである。ARTEMISは次の3つのモジュールで構成される。**CP-DLL**は各プロセスに動的にリンクされ、プロセスのチェックポイント・リスタートを行なうダイナミックリンクライブラリである。**PM**は各計算機に存在し、同一計算機上のプロセスの監視をおこなう。**GM**はCP生成時に各計算機上のPM間のタイミングをとる。これらのモジュールによりARTEMISの3つのキー技術を実現している。次章以降では、以上の3つのキー技術について順に説明していく。

3 分散CPプロトコル

通常、分散システム上では、複数の計算機上のプロセスがIPCを行ないながら処理を進める。このような環境において、各プロセスがそれぞれ独自のタイミングでCPを生成した場合、矛盾が発生し、リスタートできないケースがでてくる。

図3は3つのプロセスがメッセージ送受信を行なっている場合のCP生成のタイミングの例を示したものである。図中のCPiはある時点の3つのプロセスのCPのセット、cp-jiはCPi時点でのプロセスjのCPを示している。それぞれのタイミングにおいて、CP生成直後、障害が発生してリスタートした場合を考えると以下のようになる。

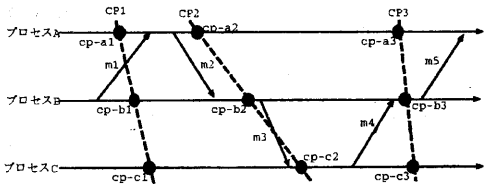

図3: 分散CP生成のタイミング例

* **CP1(NG):** プロセスBはメッセージm1を送り終わった状態で、プロセスAはメッセージm1を受ける前の状態のため、メッセージm1が失われてしまう
* **CP2(NG):** プロセスBはメッセージm3を送る前の状態で、プロセスCはメッセージm3を受けた後の状態のため、メッセージm3が2回送られることになる
* **CP3(OK):** 矛盾を起こすメッセージがないので、正しくリスタートできる

このように、プロセス毎のタイミングでCPを生成したのでは、正しくリスタートできないケースがでてくる。すなわち、"送信されたが、受信されていないメッセージ"や"送信されていないが、受信されたメッセージ"のようなメッセージが存在しないCPが一貫性のあるCPであるということができる。この問題を解決するプロトコルがいくつか提案されているが[4]、それらはメッセージ送受信を対象にしており、その他のIPC(共有メモリ、パイプ、セマフォ等)への対応は困難である。

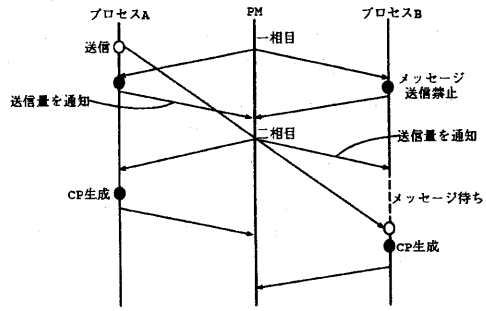

図4: ARTEMISのCPプロトコル

図4にARTEMISの分散CPプロトコルを示す。各プロセスは送信停止フラグをもち、メッセージ送信のジャケットルーチンでは送信停止フラグをチェックし、セットされていなければ送信を行い、

セットされていればリセットされるまで送信を待機する。

図4は同一計算機上のプロセスA、Bがメッセージ送受信を行なっている場合のCP生成の手順を示している。PMは一相目で同一計算機上のプロセスに以降のメッセージ送信を禁止するように伝える。各プロセスは送信停止フラグをセットし、PMに応答を返す。PMは全ての要求の応答が揃った時点で、二相目を開始する。各プロセスは二相目の要求を受けて、プロセスのCPを生成し、送信停止フラグをリセットする。すべてのプロセスのCP生成が正常に終了した時点で分散チェックポイントが成立することになる。このように、最初にすべてのプロセスのメッセージ送信を停止した後、CP生成を行なうことで、“送信されていないが、受信されたメッセージ”が存在しないことになる。また、各プロセスは二相目開始までの間、メッセージ送信以外の処理は継続できるため、分散チェックポイント生成によるプロセスへの影響を小さくできる。

このとき、図4に示すように、CP生成プロトコル開始より前にメッセージを送信して、まだそれを受信していない場合を検出するため、各プロセスは一相目の応答時に、前回のCPから現在までの間に、プロセスBに送信したメッセージの総量を共に通知し、PMは二相目の開始時にその送信量をプロセスBに通知する。プロセスBは通知された送信量を受信するまで、CP生成を遅延する。このような処理により、“送信されたが、受信されていないメッセージ”が存在しないことになり、図のようなタイミングでも正しくCPが生成できる。

通信プロトコルの主なものとして、TCP、UDPがあるが、送信量の交換を必要とするのは、TCPの場合のみである。すなわち、プロセスAとプロセスBの間での接続に関して、送信量の交換を行なう。これに対して、UDPプロトコルでは、メッセージロストの可能性を許している為、“送信されたが、受信されていないメッセージ”が存在してもよい。そのため、UDPについては、送信量の交換およびCP生成の遅延を行なわない。

図4では、同一計算機上のプロセスの例を説明したが、実際には、GMとPMとの間で複数の計算機にまたがってこのプロトコルが実行される。GMは一相目で各計算機のPMに一相目の通知をし、全てのPMからの応答がそろったところで、二相目を開始し、すべてのPMからの応答がそろったところで、分散CPが成立する。これにより、図4と同様の効果が複数の計算機環境で実現できる。

また、これまでの説明では、メッセージ送受信の場合について述べたが、ARTEMISでは、IPCを計算機間IPCと計算機内IPCに分類している。計算機間IPCとしては、ネットワークを介したメッセージ送受信があり、計算機内IPCとして、共有メモリ、ファイル共有によるデータ交換、セマフォ、パイプ等がある。計算機間IPCについては、これまで述べた方法で調停を行なう。計算機内IPCについては、CPプロトコルの一相目でアクセスを停止するのではなく、二相目でセマフォ等を利用してプロセス間で同期をとり、同一計算機上のプロセスのアクセスが停止された状態にしてCP生成を行なう。

4 分散レプリケーション技術

分散レプリケーション技術は、サーバ計算機障害に対して、CP情報を元にすぐに他の計算機上で処理を引き継ぐための技術である。DBファイル等のファイルの複製を管理すると共に、プライマリ・サーバ計算機上で動作しているサーバプロセスや共有メモリのCP情報をバックアップ計算機側に転送する。以下では主に、複製ファイルの管理方法について説明する。

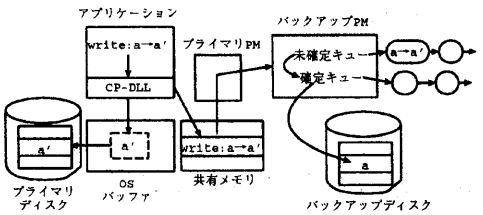

図5: 複製ファイルのサポート

図5はプライマリディスク上とバックアップディスク上にある複製ファイルの一貫性を保つための構成を示したものである。図では、アプリケーションがファイルの「a」の部分を「a'」に更新する場合の処理を示している。通常の処理中は以下のような手順になる。

1. アプリケーションがwriteを発行
2. ジャケットルーチンで、「a'」に更新するという更新情報を共有メモリに格納

3. 実際の write システムコールを発行し、ファイルの更新を行なう

共有メモリ上の更新情報は、プライマリ側の PM により順次バックアップ側に転送する。更新情報は、CP 生成までにバックアップ側に転送すればよい。バックアップ側の PM は受け取った更新情報を未確定キューにつなぐ。

CP 生成時の手順は以下のようになる。

1. 未転送の更新情報があれば、それをバックアップ側に転送する
2. バックアップ側 PM は未確定キューにつながれている更新情報を確定キューにつなぎかえる
3. 確定キューの更新情報は順次バックアップファイルに反映される。ただし、プライマリ・サーバ計算機障害により、バックアップ側に処理が引き継がれる場合は、確定キューの内容をただちにバックアップファイルに反映する。

以上の処理で、プライマリファイルの更新は、バックアップ側への転送およびバックアップファイルの更新の完了を待つ必要がないため、ファイルの複製を効率良く管理できる。

また、サーバプロセス障害のためにプライマリファイルの内容をロールバックする場合、バックアップディスク上にプライマリディスクの更新前データが存在するため、共有メモリおよびバックアップ PM の未確定キューの更新情報とバックアップディスクの更新前データよりプライマリディスクの内容をロールバックすることができる。

また、プライマリ・サーバ計算機障害で処理をバックアップ側に引き継ぐ場合、確定キューおよびバックアップディスクの内容により、CP 時のプライマリディスクの状態にできるため、アプリケーションは矛盾無くリスタートできる。

5 ジャケットルーチン

ジャケットルーチンは、プロセスが発行するシステムコールをフックし、プロセスの状態の保存 (CP 生成)・回復 (リスタート) を行う。プロセスの状態は OS に依存する部分が多いが、本稿では、UNIX[3](Solaris[4]) の場合について述べる。

[3]UNIX は X/Open の商標です。
[4]Solaris は米国 Sun Microsystems 社の商標です。

プロセス資源はアドレス空間およびレジスタセットといったプロセスから直接アクセスできる資源と、ファイル、共有メモリ、socket といった OS により提供される資源の 2 つに大別できる。アドレス空間の情報は、前回の CP 生成時から現在までに更新されたページ (ダーティページ) だけを保存する差分保存方式 [5] をとっている。レジスタセットに関しては、setjmp/longjmp により保存・回復を行う [6]。OS 資源についてはプロセスが OS 呼び出しを行うシステムコールをジャケットルーチンでフックし、必要な情報を保存・回復する。

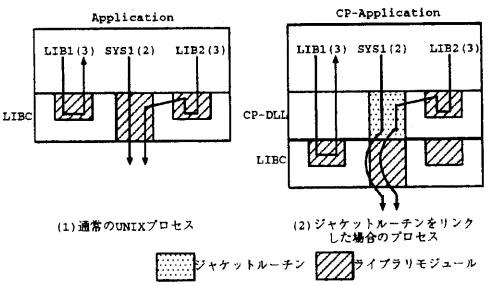

図 6: システムコールのフック方法

図 6 はシステムコールをフックするための仕組みを示した図である。通常のプロセスは図 6(1) に示すように OS をアクセスするためのシステムライブラリ (LIBC) がリンクされる。LIBC は通常、システムコール (SYS1、ex. open)、ライブラリコール (LIB1、ex. strcpy)、間接的システムコール (LIB2、ex. fopen) の 3 種類のインタフェースを提供する。

これに対して ARTEMIS では、図 6(2) に示すようにジャケットルーチンを実装した CP-DLL が LIBC より先にリンクされる。ARTEMIS では必要なシステムコールをフックするジャケットルーチンを持っており、アプリケーションがシステムコール SYS1 を発行した場合、LIBC の SYS1 ではなく、ジャケットルーチン SYS1 が呼ばれる。また、LIB2 のように間接的に SYS1 を呼び出すような関数に関しては、LIBC から LIB2 のオブジェクトを取り出し、ともにリンクする。この場合 LIB2 からの SYS1 の呼び出しは SYS1 を呼び出すように解決されるため、最終的にはジャケットルーチン SYS1 が呼び出される。

このように、ジャケットルーチンをアプリケーションにシステムライブラリより先にリンクすることにより、アプリケーションの変更を必要とせず、

そのアプリケーションに関するOS内の状態を保存することが可能になる。

6 実装

現在、ARTREMISはUNIX (Sparc-Solaris 2.3、2.4) 上で開発を行なっている。実装を行なったシステムでは、Oracle[5]DBシステムとWebサーバとWebクライアント (Mosaic) からなる3層C/Sシステムにおいて、MosaicからDB更新を行なっている途中でサーバに障害が発生した場合に、処理がバックアップ側に引き継がれることを確認している。以降では、Mosaic等のX-Windowアプリケーションへの対応方法について説明する。

X-Windowシステムは、ユーザアプリケーションの処理を行なうXクライアントと、Xクライアントからの要求により、ディスプレイにウィンドウを表示するXサーバからなる。アプリケーションプロセスの入出力がX-Windowシステムを介して行なわれる場合、ARTEMISでもそれに対応する必要がある。対応方法としては、XサーバのCP生成を行なう方法やクライアントライブラリを変更する方法などさまざま方法が考えられる。

ARTEMISでは、xmove[6]という標準配付の中に含まれる疑似サーバをを改造し、Xクライアントのチェックポイント生成を可能にしている。

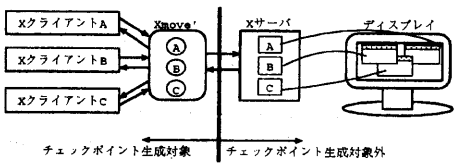

図7: xmoveによるウィンドウの状態保存

図7は、xmoveを改造したxmove'を導入した場合の構成を示したものである。Xmoveは、Xサーバ上の表示を中断/再開する機能を持っている。通常時、XクライアントからXサーバへの要求やXサーバからXクライアントへの応答はxmoveを介して行なわれるため、これらの一部をxmove内部に保存する(静的情報)。中断時には、さらにビットマップ情報等をXサーバから獲得し(動的情報)、保存しておいた静的情報とともに保存しておく。再開時には、保存しておいたXクライアントの情報をもとにXサーバにウィンドウの表示を依頼する。

この機能を利用して、xmove'ではCP生成時には、Xサーバから動的情報を獲得してから、xmove自体のCP生成を行なう。これにより、Xクライアントの情報はxmove'のアドレス空間の一部として保存される。リスタート時には、xmoveはXクライアントの表示が中断された状態と同じであるため、表示の再開の処理を呼び出す。このような処理により、X-Windowシステムを利用するアプリケーションに関してもARTEMISのチェックポイント・リスタート方式が適用可能になる。

7 おわりに

本稿では、ARTEMISの概要と3つのキー技術である分散CP技術、分散レプリケーション技術、ジャケットルーチンについて述べた。実装したシステムでは、OracleDBシステムを含む3層C/Sシステム上の複数のプロセスのCPを矛盾無く生成し、プライマリ・サーバ計算機の障害発生時には、バックアップ側で処理が引き継がれる事を確認した。

今後の課題として、非同期型CP生成アルゴリズムの導入等によるCP生成のオーバヘッドの削減、N台構成のサーバへの対応、他のOSへの対応等がある。これらの課題を解決するとともに、システム管理等のツールを整える事で、より有用な技術に発展させていきたい。

参考文献

[1] 白木原他, "高信頼化ミドルウェアARTEMISの概要とチェックポイント生成方式", 情処第54回全国大会, 1997.

[2] 佐藤他, "高信頼化ミドルウェアARTEMISの分散チェックポイント生成方式", 情処第54回全国大会, 1997.

[3] 平山他, "高信頼化ミドルウェアARTEMISの分散レプリケーション方式", 情処第54回全国大会, 1997.

[4] 真鍋他, "分散チェックポイント・ロールバックアルゴリズム", 情報処理, Vol.34, No.11, pp.1366–1467, 1996.

[5] J. S. Plank, M. Beck, G. Kingsley and K. Li, "Libckpt: Transparent Checkpointing under UNIX", USENIX Winter 1995 Technical Conference, 1995.

[6] 森山他, "利用者レベルで実現したプロセス移送ライブラリ", 情処OS研究会報告 91-OS-51, 1991.

[5] Oracleは ORACLE Corporation の商標です。

[6] xmoveはコロンビア大学の Ethan Solomita 氏らが作成したものです。