新しい高信頼サーバ計算機技術──設計思想と方式

New High-Reliability Server Technique──Design Concept and Architecture

増渕　美生　Y. Masubuchi
平山　秀昭　H. Hirayama
保科　聡　S. Hoshina

ネットワーク コンピューティングが普及するにつれ，そこで使用されるオープンサーバ計算機に対して信頼性を求める声が高まっている。当社は，このようなサーバ計算機に対して，ハードウェアとソフトウェアの両方を変更することなく，低コストで，より高い信頼性を提供することを目的とした新しい高信頼サーバ計算機技術を開発した。これにより，通常の計算機ではシステムダウンとなるような一時故障から自動的に回復し，処理を継続することができるようになる。しかも，この故障回復機構はアプリケーション プログラムに対して透過であるため，ユーザはこれを意識することなく，そのままの形でプログラムを使用できる。

As network computing becomes more widespread, achieving high reliability for the open server computers used in this environment has become a subject of increasing interest.

Toshiba has developed a new high-reliability server computer technique which requires no modification of either hardware or software, thus providing high reliability at low cost. With this technique, the server computer system automatically recovers from intermittent errors which might cause system failure in an ordinary server, and can continue to operate. Moreover, since the fault recovery mechanism is implemented transparently to the application programs, users do not need to take any special considerations when using their programs.

1 まえがき

ネットワーク コンピューティング環境では，さまざまな計算機がネットワークに接続され，多様なアプリケーションを実行している。このような環境では，これらおのおのの計算機に要求される信頼性のレベルもさまざまである。例えば，ある種のアプリケーション領域ではフォールト トレラント計算機のようにきわめて高い信頼性が要求される場合があるのに対し，別の領域では一般のサーバ計算機の信頼性で十分な場合もある。

このように多様な信頼性に対するユーザの要求の背景には，必ず費用と効果のトレードオフが存在する。すなわち，ユーザは，より少ない費用でより高い信頼性を得ようとするのである。このために，従来から，ミラーディスクやRAID (Redundant Arrays of Inexpensive Disks)，冗長化電源といった高信頼化手法が開発され，ユーザが必要とする信頼性レベルに合わせて，これらが使用されてきた。

ところが，従来は，これ以上の信頼性を得ようとすると，フォールト トレラント計算機のように高額なシステムが必要となり，ここに費用・効果の両面で大きなギャップが存在していた。当社は，このギャップを埋め，ユーザの多様な要求に対する一つの解を提供するために，新しい高信頼サーバ計算機技術を開発した。

この技術は，従来の計算機ではシステムダウンとなるような故障が発生したときに，自動的に故障回復処理を実行して処理を継続するものである。回復対象とする故障は，プロセッサ，メモリ，バスなどから成る計算機本体ハードウェアの一時故障やプロセッサの固定故障のほかに，OS (Operating System) のエラーに起因するソフトウェア故障を含む。

これらの故障回復機能は，既存のオープンサーバ計算機にハードウェアとソフトウェアから成るモジュールを付加することによって実現されるため，低コスト化が可能となる。そのうえ，アプリケーション プログラムに対しては透過であるため，ユーザの観点からは，容易に導入することができるという特長をもつ。図1に，高信頼サーバ計算機

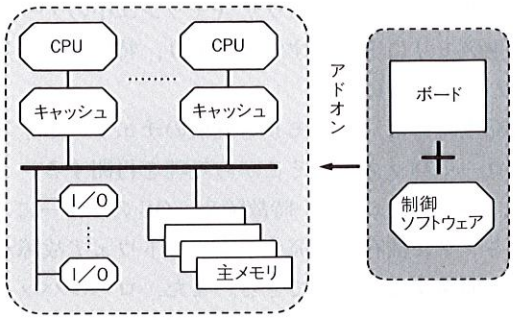

図1．高信頼サーバの実現方法　既存のオープンサーバ計算機にハードウェアとソフトウェアをアドオンすることにより，高信頼化を実現する。

High-reliability server implementation

の実現方法を示す。

なお，この技術は，フォールト トレラント計算機のようなすべての障害からの回復を目的としたものではない。ネットワーク コンピューティング環境で使用される多くの通常のオープンサーバ計算機に対して，ハードウェアとソフトウェアの両方を変更せずに，少ない費用でより高い信頼性を提供することを目的としたものである。

また，この技術は，単一の計算機の信頼性を向上させることを目的としている。これと，高可用性（HA：High Availability）技術のように複数の計算機を用いてシステム全体の信頼性を向上させる技術とを組み合わせることによって，さらに高い信頼性を実現することができる。

2 チェックポイント／ロールバック方式

図2に，この高信頼サーバ計算機技術で故障回復機能を実現するための基本機構であるチェックポイント／ロールバック方式の原理を示す。

まず，通常の処理を実行中，定期的にOSを含めたすべてのプログラムの状態を主メモリに反映させて，チェックポイントを採る。チェックポイントの採取間隔は，10 msから30 ms程度という非常に短い周期である。チェックポイント処理は，基本的に，プロセッサ キャッシュ中のダーティ データを主メモリに書き戻す処理であり，特別なメモリ領域を使うわけではない。

障害発生時には，主メモリを直前のチェックポイントの状態にロールバックし，そこから処理を再開する。これによって，ハードウェアの一時故障や，OSのエラーにより処理タイミングに依存して発生するソフトウェア故障からシステムを回復することができる。また，ロールバック時に故障したプロセッサを切り離して縮退運転することによって，プロセッサの固定故障にも対応できる。

これらの機能は，すべてハードウェアおよびOSによって実現され，アプリケーション プログラムからは完全に透過である。

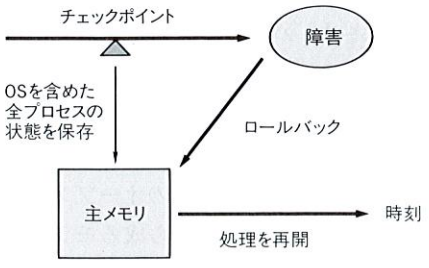

図2．チェックポイント／ロールバックの機構　故障発生時には，主メモリを直前のチェックポイント時点の状態に戻して処理を再開する。

Checkpoint and rollback mechanism

3 ハードウェア アーキテクチャ

3.1 ビフォア イメージ バッファ

チェックポイント時に主メモリ上に書かれたデータは，故障回復処理に備えるため，次のチェックポイントまで保存されなければならない。このための手法としては，①プロセッサ キャッシュから主メモリへの書出しを次のチェックポイントまで待たせる方法[1]と，②主メモリへのライトの際に更新前のデータをバッファに退避しておく方法[2]とがある。

このうち，①ではダーティ データの書出しを制限する特殊なプロセッサ キャッシュが必要になるが，通常のサーバ計算機ではこのような機能はサポートされていない。

また，②は一般にリカバリ キャッシュと呼ばれる手法であるが，従来の方式ではデータ退避機構が主メモリとプロセッサの間に設けられていた。しかし，現在のサーバ計算機では主メモリとプロセッサが一体となって構成されており，このような構成を採ることはできない。

これらに対して，当社は，現在のマルチプロセッサ型サーバ計算機のハードウェアを変更することなく，プロセッサバス上にデータ退避用バッファ（BIB：Before Image Buffer）を付加することにより，リカバリ キャッシュの機能を実現する手法を開発した。故障発生時には，BIBの内容を主メモリに書き戻すことでロールバックを実現する。

3.2 先行バックアップ方式

原理的には，BIBの機能は，プロセッサ キャッシュから主メモリにデータを書き出す際に更新前のデータを読み出してバッファに退避することで実現できる。ところが，この方式では，チェックポイント時にメモリアクセスが集中して発生することになるため，効率が悪い。

これに対して，当社は，プロセッサ キャッシュ上でデータが更新される際に，更新前のデータを主メモリから読み出して退避する先行バックアップ方式を開発した。この方式では，プロセッサ キャッシュの一貫性を保持するためにバス上に出力されるトランザクションを利用する。

一般に，プロセッサ キャッシュのデータ一貫性を保持するための方式として，各種のプロトコルが知られているが，ここでは，代表的なMESIプロトコルを用いる場合を例として取り上げる。

このプロトコルでは，キャッシュの各ブロックは，M（Modified），E（Exclusive），S（Shared），I（Invalid）の4状態によって管理されるが，BIBは，ブロックがE状態を取らないように制御する。これによって，チェックポイント後，あるキャッシュブロックに対してストア命令が実行されたときには，必ずバス トランザクションが発行されることが保証される。BIBは，このトランザクションをトリガにして，次のようにバックアップ処理を行う。

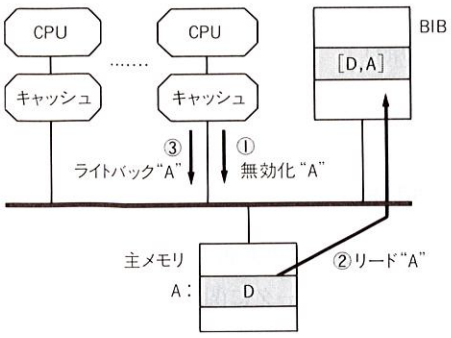

図3．S状態ブロックへのストア　キャッシュブロックへのストアが実行されると無効化トランザクションが発行されるので，バックアップのためのリードを発行して更新前データをBIBに格納する。
Store to "S" state block

(1) S状態に対するストア（図3）　プロセッサは，バスに無効化トランザクションを発行する。BIBは，これに対してリードトランザクションを発行して更新前データ(D)を読み出し，アドレス(A)とともに格納する。

(2) I状態に対するストア（図4）　プロセッサは，キャッシュブロックを主メモリから読み出すためにリード&無効化トランザクションを発行する。このトランザクションは，データの読出しを伴うため，BIBは新たなトランザクションを発行せず，読み出されたデータをバスから取り込んでアドレスとともに格納する。

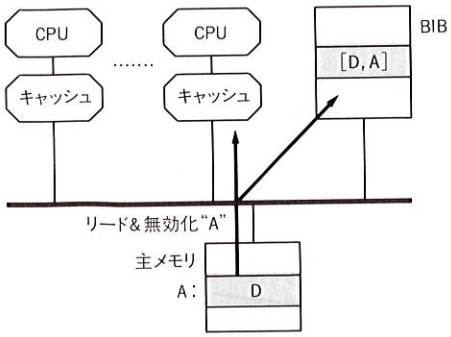

図4．I状態ブロックへのストア　キャッシュブロックへのストアが実行されるとリード&無効化トランザクションが発行されるので，これにより読み出されたデータをBIBに格納する。
Store to "I" state block

(3) M状態に対するストア　この場合はトランザクションが発行されないが，チェックポイント後，最初にM状態に遷移した時点でバックアップが取られているため，問題はない。

3.3 チェックポイント処理のサポート機構

チェックポイントを採る際には，キャッシュ中のダーティデータを主メモリに書き戻す必要がある。この処理を高速化するために，モディファイドブロックテーブル（MBT：Modified Block Table）というハードウェア機構をBIBとともにもつ。

MBTは，バス上に発行されるトランザクションを観測して，キャッシュ上でどのブロックがM状態にあるかを記録し，管理する。チェックポイント処理時には，このテーブルを参照し，M状態のキャッシュブロックに対して主メモリへの書戻しを起動するようなバストランザクションを発行する。

これらの処理をハードウェアで並行して実行することによって，高速化を実現している。

4 ソフトウェアアーキテクチャ

4.1 チェックポイントの採取

障害発生時にロールバックするため，定期的にOSを含めたすべての状態を主メモリに反映させて，チェックポイントを採る。その手順は次のとおりである。

(1) プロセッサのコンテクストを保存する。
(2) 保存されたコンテクストも含め，キャッシュ中のダーティデータを主メモリに書き戻す。
(3) BIBをクリアして更新前データを捨てる。

これらの処理は，チェックポイント処理専用プロセスによって実行される。図5にこの専用プロセスの動作を示す。この専用プロセスは，最高位の優先度を与えられており，通常はスリープしている。チェックポイント採取の要求が発生すると，その要求処理の中で専用プロセスを走行可能状態にする。最高優先度を与えられた専用プロセスは，速やかにディスパッチされ，チェックポイント処理を実行する。この処理が終了すると，専用プロセスは再びスリープ状態に戻る。

マルチプロセッサシステムでは，このプロセスがプロセッサの数だけ存在し，各プロセッサに一対一でバインドされている。チェックポイント処理を行う場合には，すべてのチェックポイントプロセスが起動される。全プロセッサ

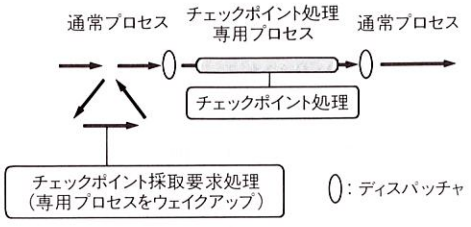

図5．チェックポイント専用プロセスの動作　チェックポイントの採取要求があると，スリープしていた専用プロセスを走行可能状態にする。チェックポイント処理を実行した後，再びスリープ状態に戻る。
Action of dedicated checkpoint process

は，同時にチェックポイント処理を実行し，最後に同期を取ってからBIBをクリアして処理を終了する。

なお，チェックポイントは自動的に採られるため，アプリケーション プログラムでこれを意識する必要はない。

4.2 二相チェックポイント

チェックポイント処理は，これを実行している間，通常処理を実行できないため，性能上のオーバヘッドとなる。この時間の大半はキャッシュからダーティ データを主メモリに書き戻す処理に費やされている。当社は，この時間を短縮するために，二相チェックポイントと呼ぶ手法を開発した。

これは，データの書戻し処理を次のような二つの相に分けて実行する手法である。

(1) 第一相　通常処理と並行して，キャッシュ中のダーティ データを主メモリに，順次書き戻す。

(2) 第二相　通常処理を停止して，書戻しだけを実行する。

第二相で書き戻されるデータは，第一相の間に新たに書換えられたものだけなので，データ量が少なくて済み，オーバヘッドが軽減される。図6に，従来のチェックポイント方式と二相チェックポイント方式の動作の比較を示す

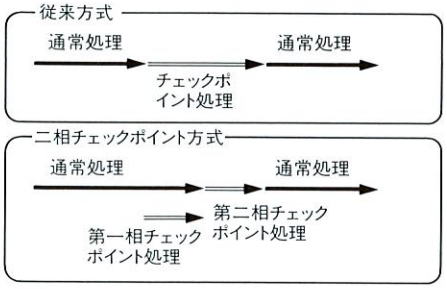

図6．従来方式と二相チェックポイント方式の比較　通常処理と並行して第一相のチェックポイント処理を実行するため，第二相に要する時間を短縮できる。

Comparison between one-phase and two-phase checkpoint

4.3 故障からの回復

障害発生時には，主メモリを直前のチェックポイントの状態にロールバックし，そこから処理を再開する。その基本的な手順は次のとおりである。

(1) キャッシュをクリアする。

(2) BIBに保存されている更新前データを主メモリに書き戻す。

(3) 書き戻された主メモリ上のデータを用いて，プロセッサのコンテクストを復元する。

これによって，プロセッサと主メモリはチェックポイントの状態に戻すことができた。ところが，入出力処理に関しては，これだけでは解決できない問題がある。次章ではこの問題を説明し，これに対して当社が開発した手法について述べる。

5 入出力処理

5.1 ロールバック時の矛盾

図7に，ディスクデータに関してロールバック時に矛盾が生ずる例を示す。ここでは，次のような状況を仮定する。

(1) ある時刻においてチェックポイントを採った後，ディスクのあるブロックからデータ“X”を読み出す。

(2) この後，同じブロックにデータ“Y”を書き込む。

(3) この後で障害が発生し，直前のチェックポイントにロールバックして，処理を再開する。

(4) このとき，再度ディスクを読み出すと，本来データ“X”が読まれなくてはいけないのに対して，データ“Y”が読み出されてしまう。

この矛盾は，入出力装置の状態がプロセッサや主メモリとは異なってロールバックしないために生ずる。

これに対して，ディスクの場合は，主メモリと同様に，書込み前に更新前のデータを読み込み，次のチェックポイントまで保存しておく，という方法が考えられるが，性能上好ましくない。

また，ネットワーク接続された他の計算機に対してデータを送信した場合には，その計算機はデータを受信して処理を進めてしまうため，ロールバックは根本的に不可能となる。

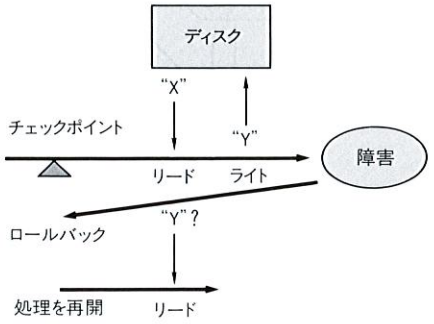

図7．故障回復に関する入出力処理の問題点　ロールバック時に，ディスクデータに関して矛盾を生ずる問題を解決する必要がある。

I/O recovery problem

5.2 入出力処理の遅延

前節で述べた問題を解決するため，次に示すように入出力処理を遅延させる。

(1) 入出力装置への出力処理が要求された場合，その処理を直ちに実行せず，次のチェックポイントまで実行を保留しておく。

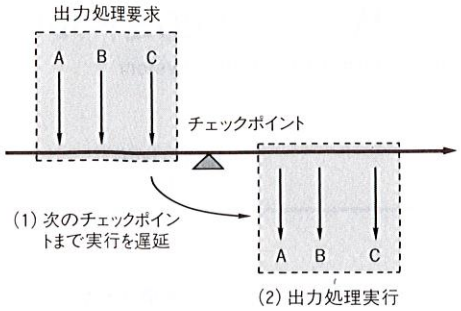

図8．入出力処理の遅延　出力処理要求は，次のチェックポイントまで実行を待たせることによって，矛盾のない入出力処理を実現する。
Deferred I/O request processing

(2)　次のチェックポイントを経過すると，システムはそれ以前の状態にロールバックされることがなくなるので，それまで保留されていた出力処理を実行することが確定する。

このため，図8に示すように，入出力装置への出力処理要求を次のチェックポイントまで遅延させてから実行すれば，プロセッサや主メモリのロールバックと矛盾のない入出力処理を実現することができる。

5.3　入出力装置の状態復旧

障害が発生した時点では，入出力装置はどのような状態にあるかわからない。例えば，入出力装置から主メモリへのデータ転送処理中であったとすると，故障回復処理を実行して直前のチェックポイント時点の状態に戻った主メモリの内容を上書きする可能性がある。この問題を解決するために，ロールバック処理を実行する前に，入出力装置をリセットして初期化する。

これによって，上で述べたような問題はなくなる。ところが，初期化されたことによって，入出力装置が直前のチェックポイント時点とは異なった状態になってしまうという問題が新たに生ずる。このため，入出力処理に伴って入出力装置の状態を管理するソフトウェア機構を設け，初期化後に状態を復旧できるようにしている。

6　あとがき

既存の計算機にハードウェアとソフトウェア モジュールを付加することによって，従来ではシステムダウンとなるような故障が発生したときに，自動的に故障回復処理を実行してシステムを継続運用する高信頼サーバ計算機技術を開発した。さらに，評価機を開発して機能評価を行うとともに，ベンチマーク プログラムを用いた性能評価を実施した[3],[4]。これにより，性能上のオーバヘッドは10％程度と，実用範囲にあることがわかった。

この技術は，オープン化時代のサーバ計算機をターゲットとした新しい信頼性向上技術である。

文　献

(1) P. Bernstein : Sequoia : A fault-tolerant tightly coupled multiprocessor for transaction processing, IEEE COMPUTER, 21, 2, pp.37-45 (1988)
(2) P. Lee, et al : A Recovery Cache for the PDP-11, Digest of papers of the 9th Annual International Symposium on Fault-Tolerant Computing, pp.3-8 (1979)
(3) 平山秀昭，他：QRMを備えた高信頼サーバ，電子情報通信学会研究会FTS96-56，pp.65-72 (1996)
(4) Y. Masubuchi, et al : Fault Recovery Mechanism for Multiprocessor Servers, Digest of papers of the 27th Annual International Symposium on Fault-Tolerant Computing, (1997)

増渕　美生　Yoshio Masubuchi
研究開発センター 情報・通信システム研究所主任研究員。計算機アーキテクチャ・高信頼システムの研究開発に従事。情報処理学会，電子情報通信学会，ACM，IEEE会員。
Communication & Information Systems Research Labs.

平山　秀昭　Hideaki Hirayama
情報・通信システム技術研究所 開発第一担当開発主務。OS・フォールトトレラントシステムの研究開発に従事。情報処理学会，電子情報通信学会会員。
Information & Communications Systems Lab.

保科　聡　Satoru Hoshina
情報・通信システム技術研究所 開発第一担当開発主務。OS・フォールトトレラントシステムの研究開発に従事。
Information & Communications Systems Lab.