Vol.2011-ARC-195 No.17
Vol.2011-OS-117 No.17
2011/4/13

仮想化技術を用いた 二重系動作環境同期機構

國分俊介[†] 金城聖[†] 鶴薫[†] 飯塚剛[†]

近年，汎用サーバと仮想化技術を用いて仮想マシンの動作状態ごとソフトウェア処理にて同期を行うことで高信頼システムを実現するソフトウェア FT 技術が注目されている．
本稿では，仮想化レイヤでサーバの動作状況に応じて同期周期を変動させながら待機系と同期を行うことにより，仮想マシン上の業務アプリケーションの実行性能への影響を抑えて，サーバ障害発生時に高速に待機系へ系切替可能な二重系動作環境同期機構について述べる．

Dynamic Redundancy Control Method in Virtual Machine Environment

Shunsuke Kokubu[†], Akira Kanashiro[†], Kaoru Tsuru[†] and Tsuyoshi Iizuka[†]

Recently, the software fault tolerant technology (software FT) is paid to attention for achieving a dependable computing system. The software FT synchronizes each state of virtual machine between an active server and a standby server using virtualization technology.
In this paper, we propose an dynamic redundancy control method that is possible to synchronize each state of virtual machine while suppressing effects on business applications by changing the synchronous cycle according to the server load. And it is possible to take-over at high speed to a standby server at active server machine breakdown.

1. はじめに

企業業務の中核となる企業情報システムや社会インフラを支える情報システムにおいては，高い信頼性が求められている．通常，高信頼なシステムを構築する際には，フォールトトレラントサーバ（以下，FT（Fault Tolerant）サーバ）や高可用クラスタソフトウェアによるサーバ二重化といった冗長化技術を用いる．

一方，多数の汎用サーバで構成されたシステムにおいて，複数のサーバを仮想化技術によって集約することにより，コスト低減やリソースの有効活用を行う方法が注目されている．このような集約されたサーバ環境での信頼性向上のためには，物理サーバ，または，仮想マシン単位で高可用クラスタソフトウェアによる冗長化を行うことで可用性向上を行っている．これにより，物理サーバ，または，仮想マシン上で障害が発生しても，もう一方のサーバで動作を継続することができる．

しかし．仮想化技術によってサーバ機能が 1 台のサーバに集約されるにつれ，物理サーバ故障時の影響が増大しており，従来の二重化以上のシステム可用性が求められている．従来の高可用クラスタソフトウェアによる冗長化では，運用系で障害が発生した場合の切替には，待機系での業務引継ぎ処理等により時間がかかっていた．また，切替時間を短くする為には，業務アプリケーションと密接に連携した独自の構成制御機能が必要であった．

このような背景のもと，仮想化技術を応用し，ソフトウェア処理で仮想マシンの実行状態を逐次待機系物理サーバへコピーしておき，物理サーバ停止時には業務アプリケーションに系切替を意識させることなく，高速に待機系物理サーバへ系切替可能とするソフトウェア FT 技術が注目されている．

本稿では，このソフトウェア FT 技術をベースに，サーバの状況に応じて同期周期を動的に変更させることで，効率的に，かつ，仮想サーバ上で動作する業務アプリケーション処理への影響を少なくし，また，仮想サーバ上の高可用クラスタ等のアプリケーションから系切替等の指示が可能な二重系動作環境同期機構を開発したので報告する．以降，2 章で本稿に関連する技術一般について触れ，3 章で技術課題について述べる．さらに，4 章で本稿で提案する二重系動作環境同期機構について詳細に説明し，5 章で実装，及び，評価内容について述べる．6 章で評価結果に基づく結論を述べ，最後に 7 章でまとめる．

2. 関連技術

2.1 FT サーバ

従来，社会インフラ向けシステム等の高い信頼性を要求されるシステムを構築するためには，FT サーバ等の耐故障性を向上させる専用機構を備えたサーバが用いられてきた．通常の FT サーバではメモリ，CPU，ストレージ，NIC，電源等のハードウェア各部位が冗長化されており，常に運用系のハードウェアコンポーネントと待機系のハードウェアコンポーネント間で同期が取られており，片方のハードウェアコンポーネ

[†] 三菱電機株式会社 情報技術総合研究所
Mitsubishi Electric Corporation, Information Technology R&D Center

Vol.2011-ARC-195 No.17
Vol.2011-OS-117 No.17
2011/4/13

ントが故障しても，もう片方のハードウェアコンポーネントで動作が継続する．しかし，特殊なハードウェアを用いて冗長化を行うため，汎用 PC サーバと比べコストがかかる等の課題がある．

2.2 ソフトウェア FT

近年の仮想化技術の発展に伴い，物理サーバ間で仮想マシンのメモリやディスクなどの動作状態を同期させることで，一方の物理サーバが停止しても，他方の物理サーバ上の仮想マシンが動作を継続する技術が注目されている．本稿では，このような技術を総称してソフトウェア FT と呼ぶ．ソフトウェア FT はソフトウェア処理のみで実現しており，FT サーバのように特殊なハードウェアは必要ないため，低コストで耐故障性を向上させることが可能となる．また，ソフトウェア FT 環境では，仮想マシンそのものを同期しておくので，仮想マシン上で動作する OS やアプリケーションに依存せずに障害発生時に系を切り替えることが可能となる．ソフトウェア FT の実装例としては，例えば，オープンソースの Remus[1]や Kemari[2][3][4]，市販製品の VMware FT[5]，everRun FT[6]等がある． ソフトウェア FT における同期方式にはいくつかの方式がある（表 1 参照）．

表 1　各仮想化ソフトウェアで対応する同期方式一覧

VMware FT	everRun FT	Remus	Kemari
ログ方式：運用系でのイベント処理を FT ログとして待機系へ送信し処理する．	ロックステップ方式：運用系と同一の処理を待機系でも行うことで同期する．	周期同期方式：一定周期ごとに運用系のメモリ/ディスク等のデータを待機系へ送信し反映させる．	I/O イベント同期方式：ある I/O イベント発生ごとに運用系のメモリ/ディスク等を待機系へ送信し反映させる．

各ソフトウェア FT に関しては，Remus のようにあらじめ設定された時刻ごとに同期処理を行う（a）周期型同期方式と，VMware FT/everRun FT/Kemari のようにある特定処理や I/O 処理などのイベントごとに同期処理を行う（b）イベント型同期方式に大別できる．

3. ソフトウェア FT における課題と解決策

3.1 同期処理に関わるオーバヘッド

2.2 節で紹介した各ソフトウェア FT において，（a）の方法では，同期周期を短くすると同期処理自体のオーバヘッドが大きくなり，同期対象の仮想マシン処理性能に影響を及ぼしてしまう．逆に，同期周期を長くすると，その分の同期に関わるデータ量が増大してしまうため，同期中の仮想マシン上の I/O のブロック時間がその分大きくなってしまい，結果として仮想マシンの処理性能に影響を及ぼしてしまう．一方，（b）の方法では，同期処理が発生するイベント量に左右されている．例えば，同期対象仮想マシン上で頻繁にディスク書き換えが発生する業務アプリケーションが動作している場合，その都度同期処理が発生してしまい，その仮想マシン自体の処理性能に影響を及ぼす可能性がある．

3.2 同期による共連れ停止

ソフトウェア FT を利用することで，ハードウェアの故障による系切り替え時間は高可用クラスタソフトウェアを利用するより短くなる．しかし，ソフトウェア FT では，仮想マシン上で動作する OS やアプリケーションの障害ハングアップへは対応できない．これらに対応するためには，仮想マシン，もしくは，管理用ホスト上に高可用クラスタソフトウェアが必要となる．さらに，ソフトウェア FT の待機系物理サーバとは別に，高可用クラスタのスタンバイ状態として動作するサーバが必要となる．このスタンバイ系による復旧では，高可用クラスタが仮想マシン上のソフトウェア，もしくはゲスト OS の再起動を行うため，ソフトウェア FT による復旧に比べて切替時間が長くなる．このとき，仮想マシン上の高可用クラスタソフトウェアで検出する障害として，ソフトウェア障害の他に、OS レベルでは検出できない障害がある．例えば，ディスクやネットワーク I/O のタイムアウト時間は，通常，OS レベルでは長く設定されているが，社会インフラ向けシステム等の制御系システムでは，システムとしての応答時間を保証するために高可用クラスタソフトウェアでタイムアウト時間を短く設定し，これより応答時間が長い場合に異常と判定し，運用系を停止することにより系切替を行うようになっている．このような OS レベルで異常ではなくてもシステム的に異常な状態は，ソフトウェア FT により待機系に同期され，実際には待機系のハードウェアに異常がないにもかかわらず，運用系・待機系ともに停止してしまう（本稿ではこれを「共連れ停止」と呼ぶ）．このため，高可用クラスタのスタンバイ系サーバがない場合はシステムが完全にダウンし，また，スタンバイ系サーバがあったとしても復旧までの時間が長くなってしまう．

3.3 解決策

本稿では，周期同期方式の Remus（Xen[7]）をベースとして，同期対象仮想マシンの動作状況に応じて同期周期を動的に変動させることで 3.2 節の課題を解決する同期周期動的制御機能を開発した．また，同期開始/停止や系切替を制御可能なインタフェースを提供し，高可用クラスタソフトウェアや仮想マシン上の業務アプリケーションの処理タイミングで同期制御を行うことで 3.2 節の課題を解決するアプリケーション連携機能を開発した．本稿では，これら機能を含む二重系動作環境同期機構を開発したので，4 章で詳細に説明する．

Vol.2011-ARC-195 No.17
Vol.2011-OS-117 No.17
2011/4/13

4. 二重系動作環境同期機構

4.1 全体構成

本稿で提案する二重系動作環境同期機構では，ベースとなるソフトウェア FT 機構として Xen4.0 から導入された Remus を利用する．Remus の構成を図 1 に示す．

Remus は，固定周期による動作環境の同期を行っている．Remus は同期開始指示時にパラメータとして同期周期を渡す仕様となっており，周期を渡さない場合は，デフォルト値である 200msec で同期周期の設定がなされる．Remus では，運用系側と待機系側でハードビート通信を行うことで互いの生死確認を行っており，このハートビート通信が途切れることで，系切替を行う．

Remus では，運用系仮想マシン のメモリ情報を取得する際に，一時的に運用系仮想マシンを停止する．運用系仮想マシンの停止中にメモリ情報を取得し，取得完了後に運用系仮想マシンの動作を再開する．Remus では，系切替時に I/O（特にネットワーク I/O とディスク I/O）との不整合が生じないように，同期がコミットされるまでの間，運用系側にて各 I/O をブロックする．ネットワーク I/O に関しては，運用系側にてバッファリングを行い（図 1 の①），ディスク I/O に関しては，運用系側では，そのままディスクに書き出し，待機系側では I/O 要求のバッファリングを行っている（図 1 の②）。

この Remus の動作を踏まえた二重系動作環境同期機構の全体構成は図 2 のようになる．二重系動作環境同期機構は Xen の管理ホスト（Dom0）上に配置するため，仮想

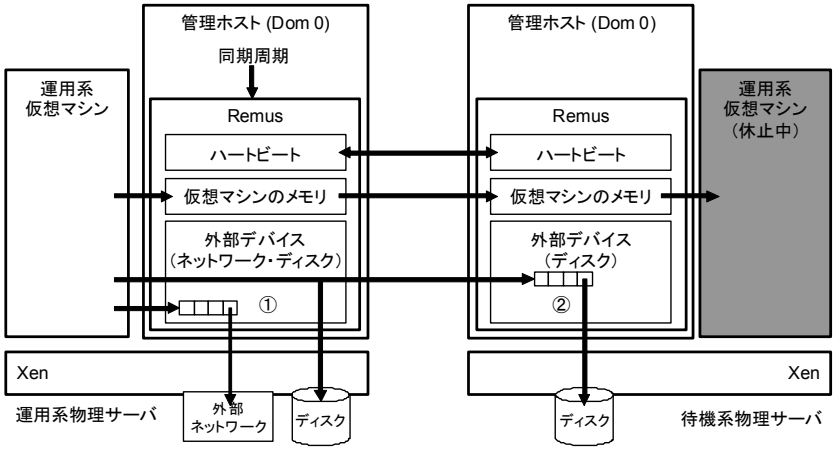

図 1 Remus 構成

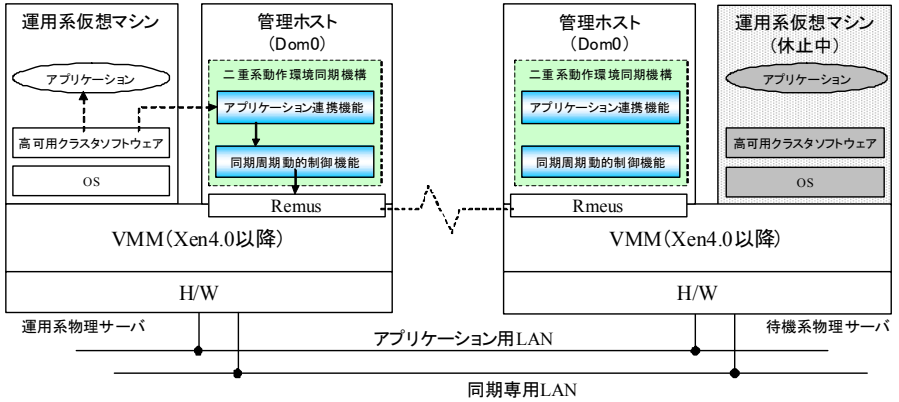

図 2 二重系動作環境同期機構全体構成

マシン上へは本機構に関わる機能を追加する必要がなく，従来の仮想マシンをそのまま使用できる．

本機構は大きく同期周期動的制御機能とアプリケーション連携機能から構成されており，同期周期動的制御機能は，仮想マシンの動作状態を元に仮想マシンの処理性能への影響を最小限にするような同期周期を算出し，適宜 Remus の同期周期を変更する制御を行う．

アプリケーション連携機能は，仮想マシン上の高可用クラスタソフトウェアやアプリケーションからの指示に伴い，同期周期動的制御機能の動作設定や，同期開始/停止，また，待機系側への強制系切替を行う．

以降でこれら同期周期動的制御機能とアプリケーション連携機能の詳細について述べる．

4.2 同期周期動的制御機能

4.2.1 周期制御における課題

3.1 節でも述べたが，特に本機構でベースとして利用する Remus での同期処理時の課題について整理する．

Remus では，同期コミットから次の同期コミットまでの間，ネットワーク I/O をブロッキングするため，同期周期を長く設定した場合，通信処理の遅延時間が長くなってしまう．また，Remsu へ入力される同期周期の値によっては，以下のような課題がある．

- 短い周期で同期処理を実施した場合，図 3 中②に示すように同期処理によって，図 3 中①の同期処理未実施時の処理時間と比べて処理時間が増加する．

Vol.2011-ARC-195 No.17
Vol.2011-OS-117 No.17
2011/4/13

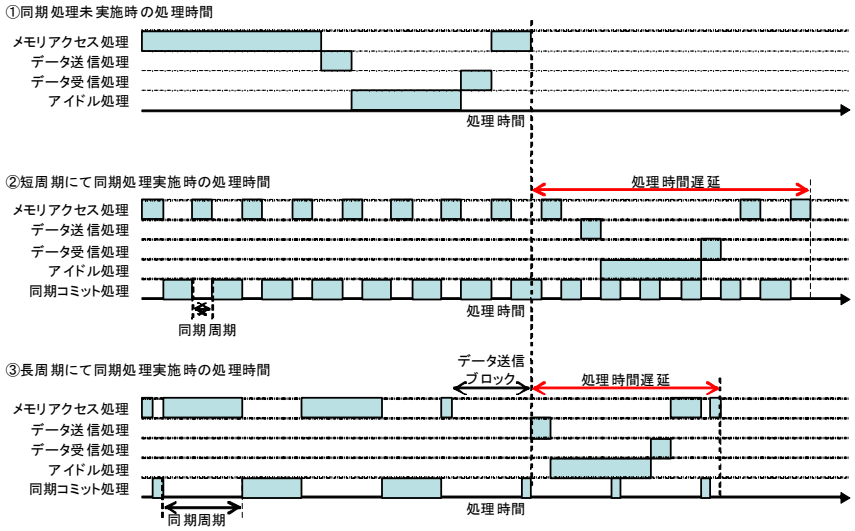

図 3　周期幅による処理時間遅延

- 長い周期で同期処理を実施した場合，図 3 中③に示すように，データ送信を伴わないメモリアクセス処理の処理時間は短くなるが，逆にデータ送信処理は，運用系と待機系間での同期処理後の通信処理内容の整合性をとるために，同期処理完了までデータ送信がブロックされるため，最長で同期周期分だけデータ送信処理に時間がかかり，図 3 中①の同期処理未実施時の処理時間と比べて処理時間が増加する．
- 通常，Remus は同期元仮想マシン動作時の変更部分のメモリダーティページを同期先へコピーすることで同期を行っている．このため，同期元仮想マシン上でメモリ書き換えが頻繁に発生する処理が動作する場合，図 3 中の③のように同期元仮想マシンのメモリダーティページサイズが増加し，メモリダーティページ取得に時間がかかり，同期コミット処理にかかる時間が増加する．

4.2.2 機能概要

同期周期動的制御機能は，上記の Remus の課題を解決し，最適な周期制御を行う機構である．同期周期動的制御機能では，同期対象仮想マシンの動作情報（ネットワーク I/O，CPU 利用率，メモリダーティページサイズ）をもとに，同期周期を算出し，算出された同期周期をもとに仮想マシンの動作に適した同期処理を実施する．メモリダーティページサイズ増加やネットワークトラフィック増加などの仮想マシンの動作情報に大きな変化が発生した場合は，再度同期周期を算出し，同期周期を変動させる．

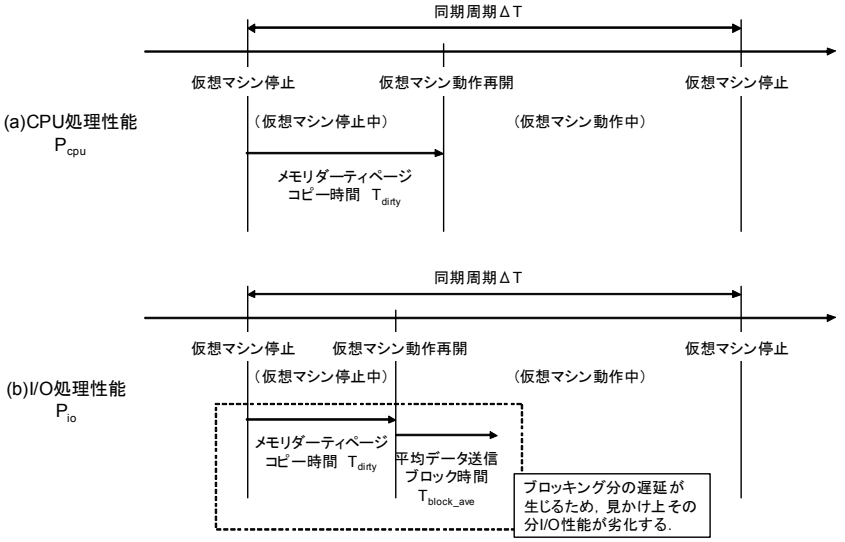

図 4　CPU 処理性能と I/O 処理性能概念図

4.2.3 同期周期演算方法

同期周期の演算方法を以下に示す．

同期中の処理性能 P_t を CPU 処理性能 P_{cpu} と I/O 処理性能 P_{io} を用いて，式①の合成関数で表す．α，β は重み付け係数である．

$$P_t = \alpha P_{cpu} + \beta P_{io} \quad \cdots式①$$

CPU 処理性能 P_{cpu} は，同期周期中において同期対象仮想マシンが動作可能な時間の割合（稼働率）[%]と定義すると，同期周期 ΔT[sec]と，メモリダーティページコピーに伴う同期対象仮想マシン停止時間を T_{dirty}[sec]として，式②のように表せる（図 4(a)）．

$$P_{cpu} = (1 - T_{dirty} / \Delta T) \times 100 \quad \cdots式②$$

このとき，メモリダーティページコピーに伴う同期対象仮想マシン停止時間 T_{dirty} は，同期処理用通信帯域を B_{mirr}[KB/sec]，書き換え発生メモリサイズを S_{mem}[KB]，同期準備に要する時間を $T_{premirr}$[sec]として式③で表せる．

$$T_{dirty} = S_{mem} / B_{mirr} + T_{premirr} \quad \cdots式③$$

Vol.2011-ARC-195 No.17
Vol.2011-OS-117 No.17
2011/4/13

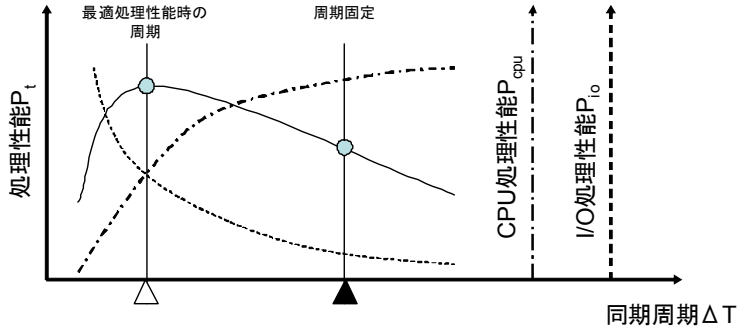

図 5　最適な同期周期の演算

また，I/O 処理性能 P_{io} は，同期周期中における実質的な仮想マシン上のネットワーク I/O [KB/sec]と定義する．同期中の仮想マシン停止やデータ送信ブロックに伴い，データ送受信に遅延が生じるため，仮想マシン上の実質的なネットワーク I/O 性能は劣化する（図 4(b)）．このため，仮想マシン用のデータ通信用帯域を B_{data}[KB/sec]，実際の仮想マシン上の単位時間当たりのネットワーク I/O を R_{vm_io}[KB/sec]，ネットワーク I/O ブロッキング時間 T_{io}[sec]とすると，式④で現せる．

$$P_{io} = (R_{vm\_io} \times B_{data} \times \Delta T) / (R_{vm\_io} \times \Delta T + B_{data} \times T_{io}) \quad \cdots式④$$

このとき，ネットワーク I/O ブロッキング時間 T_{io} は，平均データ送信ブロック時間 T_{block_ave} とメモリダーティーページコピーに伴う仮想マシン停止時間 T_{dirty} を用いて式⑤のように表す．

$$T_{io} = T_{block\_ave} + T_{dirty} \quad \cdots式⑤$$

平均データ送信ブロック時間 T_{block_ave} は，4.2.1 項の課題にもあるように，最小で 0[sec]，最大で同期周期時間ΔT[sec]の時間を要することから，本稿では平均値としてΔT/2[sec]とする．

同期周期動的制御機能では，定期的に同期元仮想マシンから B_{mirr}，S_{mem}，$T_{premirr}$，B_{data}，R_{vm_io} を取得するとき，図 5 に示すとおり式①の P_t を最大とする同期周期ΔT を算出する．

4.3　アプリケーション連携機能

4.3.1 機能概要

本稿で提案するアプリケーション連携機能は，表 2 に示すサブ機能から構成される．各サブ機能は仮想マシン上の業務アプリケーションや高可用クラスタソフトウェアから利用できるようにコマンドとして提供される．

このとき，アプリケーション連携機能で重要となるのは，3.2 節の課題にあるように，強制系切替時において共連れ停止を起こさないように系切替処理を制御することである．本処理の流れを次項に述べる．

表 2　アプリケーション連携機能（サブ機能）一覧

サブ機能名	概要
同期開始	指定の設定で仮想マシン同期を開始する．
同期停止	仮想マシン同期動作を停止する．
情報設定	初期同期周期，同期対象の仮想マシン（運用系仮想マシン），同期先物理サーバ等の情報を設定する．
情報取得	現在設定されている同期周期や仮想マシン同期動作状態，現在の仮想マシン状態等の情報を取得する．
強制系切替	仮想マシン同期動作状態を監視しつつ，強制的に同期先（待機系側）に切り替える．

4.3.2 強制系切替処理の流れ

同期処理での共連れ停止を伴わない強制系切替方法の処理の流れを以下に示す．

① 運用系の高可用クラスタソフトウェアが障害を検出し，強制系切替機能へ指示を送信する．

② 強制系切替機能は同期タイミングに応じて以下の処理を行う．

(ア) 高可用クラスタソフトウェアが障害検出前の場合
（図 6 の同期 A）
この場合は，障害検出前のため，待機系側へ移行後においては障害検出がされないため，特に処理を行う必要は無い．

(イ) 高可用クラスタソフトウェアが強制系切替指示を出す直前の場合
（図 6 の同期 B）
この場合は，待機系側へ仮想マシン動作が移行した直後に強制系切替指示が出される（共連れ停止）可能性があるため，強制系切替機能は以下の処理を行う．

① ソフトウェア FT に対して，現在同期処理が実行されているかどうかを問い合わせる．

② 同期処理が行われている場合は，自系が運用系として動作している状態での強制系切替指示と判断し，自系の仮想マシンを強制停止する．運用系仮想マシンの停止により，同期またはハートビートが途絶え，これにより，待機系側へ系が切り替わる．

Vol.2011-ARC-195 No.17
Vol.2011-OS-117 No.17
2011/4/13

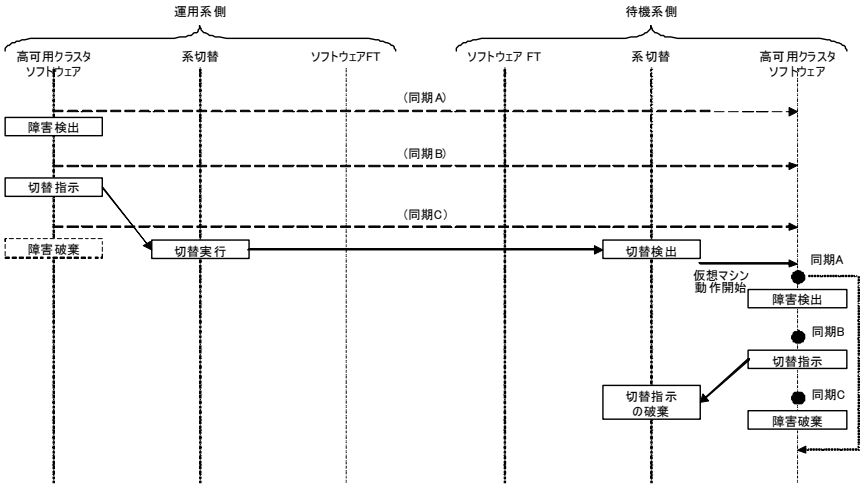

図 6　同期タイミングによる強制系切替動作

③　同期処理が行われていない場合は，仮想マシンが待機系側へ移行した後の強制系切替指示と判断し，強制系切替指示を破棄する．

(ウ) 高可用クラスタソフトウェアが強制系切替指示を出した直後の場合（図 6 の同期 C）
この場合は，待機系側の強制系切替機能へは指示は送信されず，仮想マシンはそのまま処理を続行すればよいだけであり，特に処理を行う必要は無い．

③ 高可用クラスタソフトウェアは強制系切替指示を送信後，検出した障害の状態を破棄する．以後、仮想マシン上の処理が続行される．

5. 評価

4 章で述べた二重系動作環境同期機構を仮想化ソフトウェア Xen 4.0 をベースに構築し，その動作を確認した．表 3 に使用した評価環境を示す．また，仮想マシン上には業務アプリケーションとして，クライアントからの request を受付け，処理（CPU，ディスク利用）した後に response を返す Web アプリケーションを構築した．

評価として，(1) 同期処理未実行時，(2) 定周期で同期処理実行，(3) 二重系動作環境同期機構：同期周期動的制御での同期処理実行，の 3 つのパターンでクライアントから JMeter[8]を介して業務アプリケーションへ一定の負荷がかかるような request を送信し，そのときの業務アプリケーションに対するスループットを計測した．

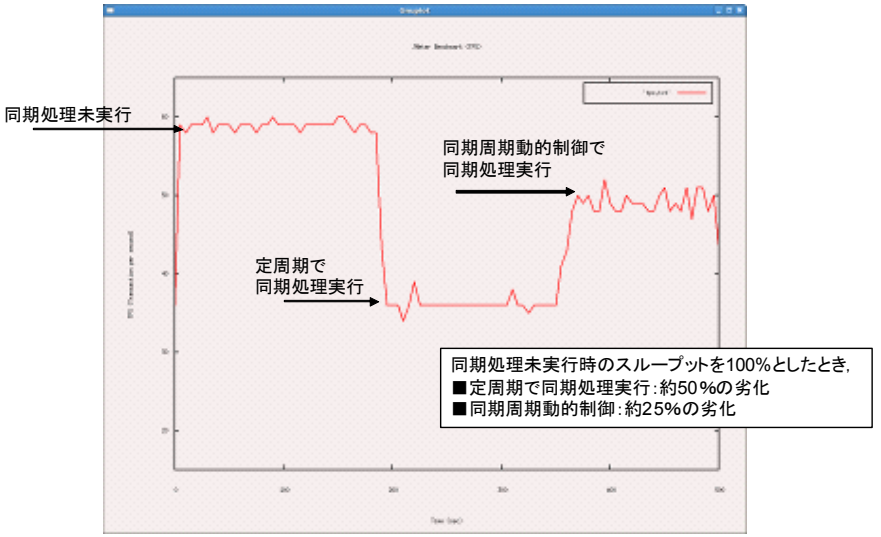

図 7 同期周期動的制御機能による処理性能変化

表 3 評価環境

サーバ	項目	詳細
物理サーバ（運用/待機）	CPU Memory Disk NIC OS	Intel Core i7-950 3.07GHz（4core） 12GB 1TB×2 Intel 82574L Gigabit Network×2 CentOS 5.4 （Xen4.0.0, kernel2.6.18-8 へ置き換え）
仮想マシン	CPU Memory Disk	仮想 CPU×1 256MB 4GB

Vol.2011-ARC-195 No.17
Vol.2011-OS-117 No.17
2011/4/13

結果は図 7 のようになり，同期処理未実行時の業務アプリケーションのスループットを 100%としたときに，一定周期での同期処理実行では約 50%劣化したが，本稿で提案する二重系動作環境同期機構の同期周期動的制御機能を用いた同期処理実行では約 25%劣化に抑えることができた．

6. 結論

従来の周期同期型ソフトウェア FT（Remus）では，同期周期を短くすると同期処理自体のオーバヘッドが大きくなり，逆に，同期周期を長くすると，その分の同期に関わるデータ量が増大してしまうため，同期中の仮想マシン上の I/O のブロック時間が大きくなってしまい，結果として仮想マシンの処理性能に影響を及ぼしてしまうという課題があった．これに対し，二重系動作環境同期機構の同期周期動的制御機能によって，仮想マシンの動作状態情報（メモリダーティページ量，ネットワーク I/O 量等）を用いて同期周期を変更させることで，仮想マシンの処理性能への影響を軽減できることが分かった（スループットに関して，約 50%劣化から約 25%劣化へ軽減）．ただし，今回は一定負荷のアプリケーションに対する評価のみのため，例えば，メモリを多く書き換えるようなアプリケーション動作時やストリーミングのようなネットワーク I/O を多く行うようなアプリケーション動作時など，様々な負荷状態において評価を実施し，同期周期演算式のチューニングを行う必要があると考える．

また，今回の評価では，一通りのアプリケーション連携機能サブ機能に関しての動作確認はできているが，仮想マシン上に高可用クラスタ環境を構築していないため，強制系切替機能についての詳細な評価ができていない．これについても，今後検証していく．

7. おわりに

本稿では，周期同期型ソフトウェア FT である Remus をベースとして，同期処理オーバヘッドや共連れ停止などの課題を解決する二重系動作環境同期機構について述べた．実際に実環境で構築・評価し，業務アプリケーションへの影響を抑えることが可能であることが分かった．

今後は，仮想マシン上の様々なアプリケーション負荷状態での性能評価や周期演算式のチューニングを進める．また，実業務環境へ本機構を適用し，アプリケーション連携機能の強制系切替の機能評価を含め，本機構の有効性を検証していく．

参 考 文 献

1) Brendan Cully, Geoffrey Lefebvre, Dutch Meyer,Mike Feeley, Norm Hutchinson, and Andrew Warfield：Remus: High Availability via Asynchronous Virtual Machine Replication, USENIX NSDI'08 (2008)
2) 田村, 柳澤, 佐藤, 盛合：Kemari：仮想マシン間の同期による耐故障クラスタリング, 情報処理学会論文誌コンピューティングシステム, Vol.3, No.1, pp.13-24(2010)
3) 大村, 田村, 吉川, フェルナンド バスケス, 盛合：KVM を利用した耐故障クラスタリング技術の開発, 情報処理学会研究報告, Vol.2010-OS-115, No.20, pp.1-8(2010)
4) Kemari，http://www.waza.jp/kemari/
5) VMware FT, http://www.vmware.com/
6) everRun FT, http://www.marathontechnologies.com/
7) Xen, http://www.xen.org/
8) JMeter, http://jakarta.apache.org/jmeter/