情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2011-ARC-195 No.25
Vol.2011-OS-117 No.25
2011/4/14

仮想計算機の同期保存・復元手法

松尾英治[†1]　花崎芳彦[†1]
伊藤孝之[†1]　飯塚剛[†1]

汎用サーバを用いて高信頼システムを実現する場合，クラスタミドルウェアなどのソフトウェアによる冗長化を行なう必要がある．このようなシステムの障害試験においては，試験毎に系間の構成制御を行なう必要があり，試験期間の長期化が課題となる．本稿では，複数の仮想計算機を同期して保存・復元することで，システムに異常をきたすことなく試験開始状態へ復元し，試験時の構成制御に要する時間を短縮する手法について述べる．

A Synchronized Saving and Restoring Method for Clustered VMs.

Hideharu Matsuo,[†1] Yoshihiko Hanazaki,[†1]
Takayuki Ito[†1] and Tsuyoshi Iizuka[†1]

High availability cluster systems are implemented by cluster middleware. If engineers test machine failures on the cluster, they will have to recover cluster middleware's mode in all relevant machines at each test case. This operation takes a long time and prolongs a test period.

In this research, we propose a synchronized saving and restoring method for clustered virtual machines. It restores virtual machine's states without split-brain in cluster middleware, and shortens the test period.

1. はじめに

本稿では，高信頼システムの障害試験に要する時間を短縮する手法として，仮想計算機の保存・復元を同期して行い，システムに異常をきたすことなく試験開始状態へ復元する手法について述べる．

以下，2 章で本手法を提案するに至った背景と課題を示し，3 章で関連研究について触れる．次に 4 章で本手法の概要を，5 章で実現方式を，6 章で評価を，8 章で今後の課題を示す．

2. 研究背景

近年 IT システムの開発/構築コスト低減に対する要求が高まっており，高信頼化と低コストをいかに両立させるかが課題となってきている[1)]．これに対して，特殊な故障防止機構を備えたメインフレーム・フォールトトレラントサーバなどを用いるのではなく，汎用サーバを用い，ソフトウェア冗長化により高信頼システムを実現する方式が普及しつつある．信頼性を重要視する社会インフラ向けシステムも例外ではなく，汎用サーバを用いる方式へと変化しつつある．

汎用サーバを用いて高信頼システムを実現する場合，クラスタミドルウェアなどのソフトウェアによる冗長化を行い，故障検知/系切り替えなどの故障対策処理を実現する．このようなシステムでは，システム開発において故障に対する十分な障害試験を行い，故障対策処理が期待通りに動作を行うことを確認することが重要である．一般に障害試験では，試験項目毎に以下の作業を行なう．

(1) 故障発生と動作確認後の故障解除
(2) 故障対策処理の動作確認（故障検知/切替など）
(3) システム復旧（サーバ再起動・データ復旧など，故障によって異常状態となったシステムを正常に戻すこと）

これら故障の発生/解除及びシステム復旧などの作業においては人手を要することから，試験実施にあたって以下のような課題があった．

- 試験期間に時間を要した場合，試験でコストがかかり，狙っていたコスト削減の効果を十分に出すことができない．
- コストや期間の制限で十分な試験が実施できない．

そのため，ソフトウェア冗長化による高信頼システムでは，試験期間を短縮する技術ついても合わせて考慮する必要がある．

試験期間を長期化させる要因のひとつとして，システム復旧が挙げられる．通常，クラスタミドルウェアを用いたシステムでは，まずクラスタミドルウェアを起動し，クラスタミドルウェアの制御下でアプリケーションプログラムを起動する．このように，主系/待機系な

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

Vol.2011-ARC-195 No.25
Vol.2011-OS-117 No.25
2011/4/14

ど，クラスタとしての動作モードを確定する系間の構成制御を実施した上でアプリケーションを起動する必要があり，正常時であってもミドルウェア/アプリケーション起動に要する時間は 10 分を越えるシステムが多い．さらに障害試験では，異常状態となったデータの復旧を行なう必要がある．このためアプリケーション起動に要する時間は正常時より大きくなり，次の試験項目を行なうための状態復旧に 20 分以上を要することも少なくない．

実際，本手法の運用対象と想定している実システムにおいても，データ復旧を含めたミドルウェア/アプリケーションの起動に長時間を要しており，故障発生と動作確認という試験本来の作業に比べ，復旧作業に大きな時間をかけている．

そのため，復旧作業をいかに短縮するかが重要となる．

3. 関 連 研 究

関連研究として，仮想化ソフトウェアのスナップショット機能を用いて，計算機の動作環境をロールバックする手法が提案されている[2)]．この方法では，あらかじめ仮想計算機（VM: Virtual Machine）上にアプリケーションを構築し，VM の動作状態を保存しておく．その後，保存した状態から VM を復元することで動作異常に陥った場合の復旧時間の短縮を図る．しかし，関連研究では，クラスタミドルウェアなどのミドルウェア/アプリケーションが動作している計算機については考察がなされておらず，ソフトウェア冗長化による高信頼システムへの適用が困難である．

クラスタミドルウェアは，適時計算機間で通信し合い，お互いの状態を監視している．そのため，保存・復元処理のタイミングによって，復元後の動作モードが矛盾し，復元に失敗する可能性がある．これは，保存・復元処理において，計算機間で通信停止が生じることに起因する（**図 1**）．死活監視の周期が 1 秒など，短い環境において特に生じやすい．

単純なスナップショットの利用では，VM 保存時にネットワークデバイスなどが停止した後に VM 本体が停止するため，その間，VM 内部の時刻は進むが通信ができないという期間が生じる．復元時についても，VM 本体，ならびに，ネットワークデバイスの準備が整うまで同様の期間が生じる．この通信停止前，もしくは，停止期間中に死活監視が行われた場合，他系が停止したものとして，クラスタミドルウェアは自身の動作モードを変更してしまう．この結果，復元後，クラスタミドルウェアの動作モードに矛盾が生じる．加えて，この期間は，割り当てリソース量や物理計算機の状態に伴い，計算機毎に期間の長さやタイミングなどのばらつきが生じる．このような期間のずれによっても，動作モードに矛盾が生じる．

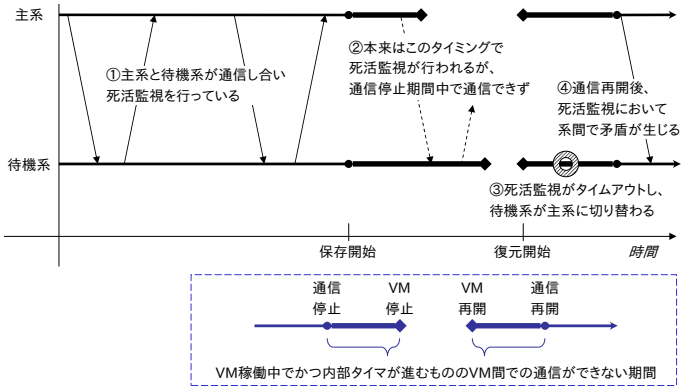

図 1 保存・復元による動作モードの矛盾
Fig. 1 conflict cluster mode by save&restore

4. 同期保存・復元手法

以上のことから，クラスタミドルウェアなどのソフトウェア冗長化による高信頼システムに対し，仮想化ソフトウェアのスナップショット機能による状態復元を適用し，試験期間の短縮を図る場合，以下のような考慮が必要となる．これら課題の解決として，仮想計算機の同期保存・復元手法を提案する．

- 保存前，復元後の間で通信に矛盾が生じないよう，保存・復元時に VM 間で行われた通信の欠落を防ぐ．
- VM 保存時の停止，ならびに，VM 復元時の再開タイミングを，1 秒の死活監視に耐えるレベルで，VM 間で同一にする．

以降，本章では，同期保存・復元手法の概要について述べる．

4.1 全 体 構 成

図 2 に本手法の全体構成を示す．本手法は，複数の仮想計算機を同期して保存・復元するものである．VM を同時に保存・復元するだけではなく，VM に割り当てられた仮想ネットワークデバイス（vNIC）のバックエンド側を保存時に切り離し，通信停止が生じないよう，バックエンド内部で通信を保存する．これにより，複数 VM の保存・復元における系間の動作モードの矛盾を抑制し，複数 VM の同期保存・復元を実現する．また，VM の状態を複数保存・復元可能にすることで，復元後の動作モードの変更作業などを不要にする．

Vol.2011-ARC-195 No.25
Vol.2011-OS-117 No.25
2011/4/14

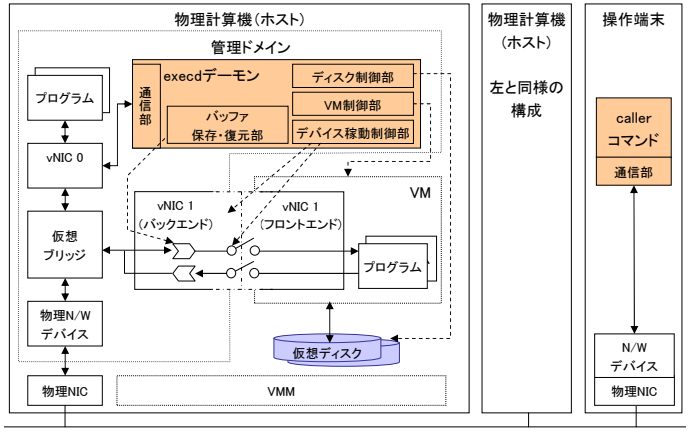

図 2 全体構成
Fig. 2 system architecture

本手法は，端末上にてユーザが呼び出す caller コマンドと，各ホスト上にて待ち受ける execd デーモンの 2 つの機構からなる．caller コマンドは，複数の execd に一斉に命令を送信する，トリガとなるプログラムである．execd デーモンは，caller コマンドからの指示に従い，自ホスト上の保存対象の VM を保存するプログラムである．execd は，VM 制御部，デバイス稼働制御部，バッファ保存・復元部，ディスク制御部から構成される．VM 制御部が保存・復元対象の VM の稼働制御．メモリ保存・復元を行ない，デバイス稼働制御部が，VM に割り当てられた仮想ネットワークデバイス（vNIC）の制御を行なう．また，バッファ保存・復元部が，vNIC バックエンド側のバッファの保存・復元を行ない，ディスク制御部が VM に割り当てられた仮想ディスクの保存・復元を行なう．

4.2 保存処理の流れ

図 3 に保存処理の流れを示す．保存は，(1) 時刻同期，(2) vNIC バックエンド切り離し，(3) VM 保存，(4) 一定時間待機，(5) vNIC バッファの保存，(6) VM 破棄，(7) ディスク保存の，計 7 ステップからなる．

(1) 時刻同期は，保存処理のタイミングをホスト間で合わすための処理である．なお，ホスト側の他アプリに支障を与えないために，補正は execd，caller のアプリケーションレベルでのみ内部的に行なう．

(2) vNIC バックエンド切り離しは，VM 保存時であっても vNIC バックエンドを稼働

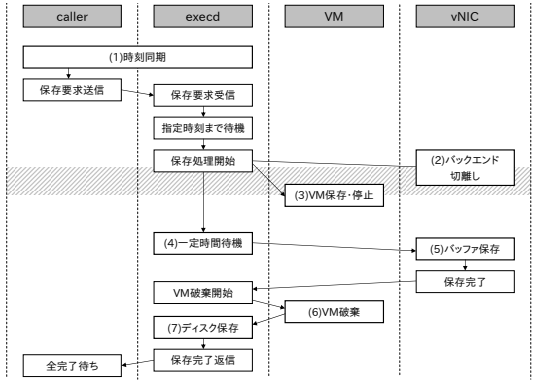

図 3 保存処理の流れ
Fig. 3 save process

させ続け，かつ，vNIC バックエンド内にパケットを閉じ込めるための処理である．これにより，VM 停止期間中における VM 間でのパケット欠落を防ぐ．

(3) VM 保存は，VM のメモリデータなどをイメージとして保存する処理である．通常の VM 保存では，メモリデータの保存後に VM を即座に破棄するが，本方式では，VM 保存後すぐには VM の破棄を行わなず，VM 停止状態のままとする．これは，VM を破棄した場合，関連する vNIC などのリソースも解放されるからである．

(4) 一定時間待機は，ネットワーク中に滞留しているパケットが相手 vNIC バックエンドに到達するのを待つ処理である．これにより，ホスト間で滞留中のパケットについて欠落を防ぐ．この期間はシステムに依存する．期間としては，クラスタミドルウェアにて指定されている死活監視の通信タイムアウトなどを想定している．

(5) vNIC バッファの保存は，vNIC バックエンドのバッファ中に溜ったパケットを保存する処理である．保存したパケットは，復元時にバッファに戻す際に利用する．

(6) VM 破棄は，停止状態にある VM を破棄する処理である．この処理により vNIC などのリソースを解放する．また，VM が利用していた仮想ディスクをアンロックする．

(7) ディスク保存は，VM が利用していた仮想ディスクを保存する処理である．スナップショットなどを用いて保存する．

4.3 復元処理の流れ

図 4 に復元処理の流れを示す．復元は，(1) ディスクの復元，(2) 停止状態での VM 復元，

Vol.2011-ARC-195 No.25
Vol.2011-OS-117 No.25
2011/4/14

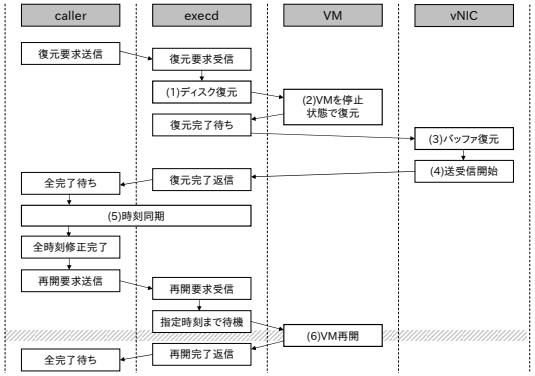

図 4 復元処理の流れ
Fig. 4 restore process

（3）vNIC バッファの復元,（4）送受信開始,（5）時刻同期,（6）VM 再開の，計 6 ステップからなる．

（1）ディスクの復元は，保存しておいた仮想ディスクを VM から参照可能にする処理である．VM の稼働には仮想ディスクが必須なため，まず最初にこの処理を行なう必要がある．

（2）停止状態での VM 復元は，保存したメモリデータから VM を復元する処理である．VM の復元後，通常は即座に VM を再開させ稼働状態とするが，本方式では，VM の復元後すぐには VM の再開を行わず，停止させた状態のままとする．これは，VM 毎に復元時間のばらつきがあり，この段階で VM 開始タイミングを合わせることが困難だからである．

（3）vNIC バッファの復元は，vNIC バックエンド中のバッファに，保存したパケットを復元する処理である．その後，vNIC バックエンドのみ稼働を行なわせる．

（4）送受信開始は，vNIC バックエンドのみ稼働を行わせ，他 VM との vNIC バックエンド間での送受信を開始させる処理である．

（5）時刻同期は，VM 再開処理のタイミングをホスト間で合わせるための処理である．この処理は，保存時の時刻同期と同様に行なう．

（6）VM 再開は，停止している VM を一斉に再開させる処理である．これにより，VM が稼働を再開し，vNIC バックエンド中に滞留しているパケットが VM へ伝達される．

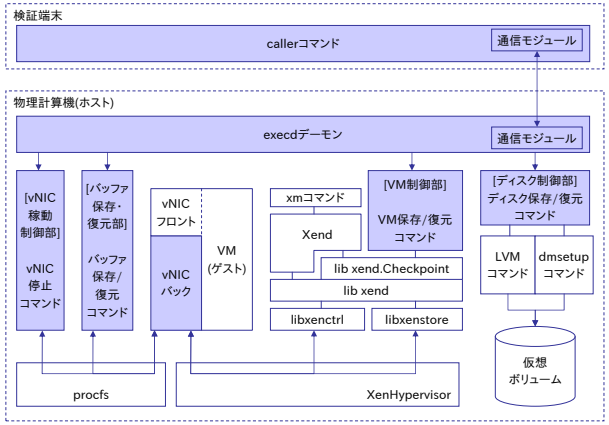

図 5 モジュール構成図
Fig. 5 module diagram

5. 実　　装

本章では，本稿で提案する同期保存・復元手法の実装について述べる．

実現においては，CentOS5(Linux[3),4)] 2.6.18-164)，ならびに，Xen[5),6)] (3.1.2-194) を用いた．また，ディスクの保存・復元においては Linux デバイスマッパー[7)] を用いた．**図 5** に本実現方式のモジュール構成図を示す．

5.1 vNIC バックエンド切り離し

vNIC バックエンド切り離しは，Xen バックエンドドライバ（netbk）に変更を加え実現する．バックエンドドライバを用いた理由は，バックエンドドライバがホスト OS 上の Linux ネットワークインタフェースとして動作しており，ゲスト OS 上へ影響を与えることなく，VM の稼動状態とは別に vNIC のみ稼動させ続けることが可能なためである．

Xen バックエンドドライバは，送受信に対応するソフトウェア割込み処理（tx_action, rx_action）において，Xen イベントチャンネルを利用し VM へ送受信情報を伝達させている．VM 同期保存･復元処理中は，この Xen イベントチャンネルを抑制し，流れているパケットをバッファリングすることで vNIC バンクエンドの擬似的な切り離しを行なう（図**図 6**）．

また，Xen は VM 保存時にバックエンドドライバの停止と解放を udev 経由で通達する．これについても，VM 同期保存において udev に対する処理を抑制することでバックエンド

Vol.2011-ARC-195 No.25
Vol.2011-OS-117 No.25
2011/4/14

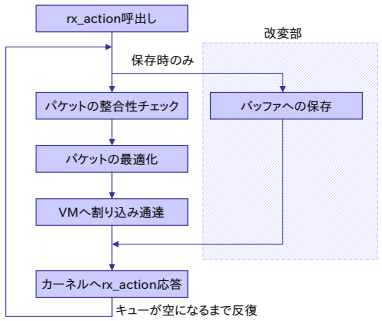

図 6 netbk における受信処理
Fig. 6 recv proccess in netbk

ドライバの停止と解放を抑制する．なお，VM 破棄前にこの抑制を解除することで，VM 破棄と共にバックエンドドライバの停止と解放が再度行われるようになる．

5.2 vNIC バックエンド制御とバッファ保存・復元

execd デーモンからの vNIC バックエンドの制御，ならびに，バッファの保存・復元には，procfs を用いる．

vNIC 作成時，vNIC 毎に procfs のエントリを作成する．procfs のエントリは，ステータス変更用エントリ（state）とバッファアクセス用エントリ（rxp），パケット数参照の 3 エントリ（rxpcnt）からなる．state は，入力された文字列を基に vNIC 内部で vNIC 切り離し状態を変更する．rxpcnt は．現在バッファ中に存在するパケット数を出力する．rxp は，現在バッファに存在するパケットを 1 つずつ出力する．また，パケットが入力された場合，それを割込みキューに登録し，ソフトウェア割込みを有効にする．

5.3 VM の保存・復元

本手法では，vNIC の制御や，開始タイミング合わせのために，VM 保存後，ならびに，VM 復元後は VM を停止させておく必要がある．

VM 保存においては，通常，Xend.Checkpoint モジュールの save 関数が用いられる．この関数では，checkpoint 引数に従い，保存後に VM の再開，もしくは，VM の破棄が行われる．本手法では，このどちらにも該当しないため，これら保存後の VM 処理そのものを削除することで対応を行なう．

VM 復元においては，通常，Xend.Checkpoint モジュールの restore 関数が用いられる．この関数では，paused 引数に従い，復元後 VM の再開が行われる．paused 引数を True にし呼び出すことで対応を行なう．

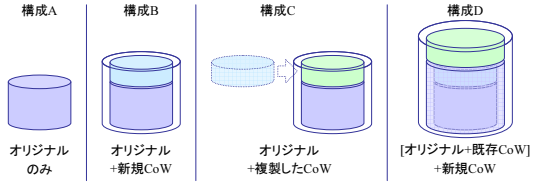

図 7 ディスク構成の種類
Fig. 7 disk composition types

5.4 ディスクの保存・復元

ディスクの保存・復元においては，Linux デバイスマッパーを用いてディスクのスナップショットを実現する．ディスクの保存・復元では，処理速度と使用容量の削減が重要となる，これは，ディスクの復元に時間を要した場合，試験時間の短縮という目的と反するからである．また，各保存状態においてディスク全体を保存した場合，保存総容量の肥大が懸念されるからである．

スナップショット技術は，ベースとなるディスクの上に Copy on Write（CoW）領域を組み合わせ，以降のディスクへの変更は CoW 領域へ行い，ベースに以前のままのデータを残す技術である．スナップショットを解除することで以前の状態に即座に戻すことが可能であり，また，以前組み合わせた CoW を再度組み合わせることで，スナップショット解除前にも即座に戻すことが可能である．

なお，スナップショットに LVM[8),9)] ではなくデバイスマッパーを直接用いた理由としては，本実装ではスナップショットを多段[⋆1]にする必要があり，LVM では実現が困難なことが挙げられる．

本手法では，次の手順でディスクの保存・復元を行なう．

- 保存のための前準備
  （1） CoW 領域を新規に作成する．
  （2） 新規に状態を構成する場合，（2）にて作成した CoW 領域を用い，dmsetup[10)] コマンドによりスナップショットを構築する（**図 7** 構成 B），既に保存してある状態を流用する場合，その保存の CoW 領域を（2）にて作成した CoW 領域へ複

⋆1 スナップショットのスナップショット

Vol.2011-ARC-195 No.25
Vol.2011-OS-117 No.25
2011/4/14

表 1 評価環境
Table 1 eval environment

サーバ	項目	詳細
物理サーバ	CPU	Xeon E5430 2.66GHz 4core
	memory	16GB
	disk	SAS 15Krpm 146GB
	OS	Centos5.4
VM	CPU	1vcpu
	memory	4GB
	disk	LVM ボリュームをマッピング

製し，同様にスナップショットを構築する（図 7 構成 C）．

- 保存処理
  （1） dmsetup コマンドによりスナップショットを解除する．
  （2） CoW 領域の名称を保存状態名へ変更．
- 復元処理
  （1） 既に復元されている場合，解除を行ない，CoW 領域を削除する．
  （2） ベースと復元対象の CoW 領域を dmsetup コマンドにより組み合わせ，スナップショットを構築する，
  （3） CoW 領域を新規に作成する．
  （4） （2）にて作成したスナップショットと新規に作成した CoW 領域を同様に組み合わせ，二段目となるスナップショットを構築する（図 7 構成 D）．

6. 評　　価

本実装を元に，復旧時間に対する評価を行った．評価は，実システムと同様のミドルウェア/アプリケーションが動作している VM を保存・復元することで行った．

評価環境を**表 1** に示す．冗長構成を組む VM4 組（計 8 台）を，2 台の物理サーバ上に，主系 VM 群，待機系 VM 群に分けて稼動させた．VM 上では，1 秒周期でクラスタミドルウェアの死活監視が行われている．

評価の結果，同期保存・復元手法を用いた場合，8VM 全ての復旧を 4 分以内で完了した．また，復元時間の大半が VM スナップショットの読み込みに伴うものであると推測されることから，VM スナップショットをそれぞれ別のディスク上に配置した場合についても評価を行った（**図 8**）．この評価の結果，保存先ディスクを別々にした場合，1 分程度に復元時間を短縮できた．

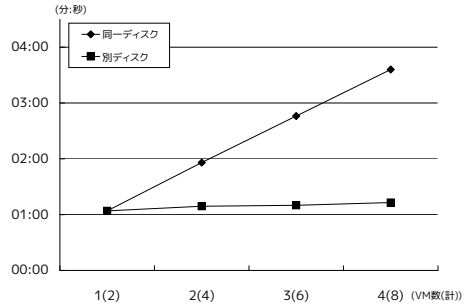

図 8 VM スナップショットの保存先ディスクによる違い
Fig. 8 relation of VM snapshot to store disk

クラスタミドルウェアの状態についても，復元後に矛盾が生じないことを確認した．また，クラスタミドルウェアの通信周期 1 秒と同じ間隔で ping 通信を行った場合であっても，VM 内において，復元後にタイムアウトが生じないことを確認した．ただし，300 ミリ秒周期で ping 通信を行った場合はタイムアウトが生じ，かつ，保存復元において VM 間で平均 400 ミリ秒程度の時刻のずれが判明した．

7. 結　　論

評価の結果，従来，障害試験においてクラスタミドルウェアの系間状態の修復に 20 分要していたものを，1 分で完了することを確認した．VM 復元時間については，VM スナップショットのディスクからの読込み時間の影響が大きく，VM スナップショット毎に別ディスクに保存することで改良できた．また，クラスタミドルウェアの状態について，復元後であっても，状態に矛盾が生じないことを確認した．

これにより，クラスタミドルウェアなどのソフトウェア冗長化による高信頼システムに対し，仮想化ソフトウェアのスナップショット機能による状態復元を実現した．本手法を試験期間における復旧処理に用いることで，試験期間の短縮が図れるものと考えられる．

なお，クラスタミドルウェアの通信周期については，現在，VM 内において 300 ミリ秒周期で ping 通信を行わせた際，復元時にタイムアウトが生じ，また，復元後，VM 間で平均 400 ミリ秒程度の時刻のずれが生じた．このため，評価において矛盾の生じないことが確認できた 1 秒以上の周期を対応可能な周期と考える．

Vol.2011-ARC-195 No.25
Vol.2011-OS-117 No.25
2011/4/14

8. 今後の課題

現在，実装において次の課題がある．

- 1 秒より短い周期で通信を行うクラスタミドルウェアへの対応
- vNIC 内バッファが溢れる可能性がある．
- 保存・復元対象外の計算機と通信を行っている場合，復元後にその通信に異常が生じる．

現在，1 秒周期の通信においてタイムアウトが生じないことを確認できた．しかし，それより短い周期で通信を行う VM についてはタイムアウトの生じる可能性が高い．この要因として，同期保存・復元における内部処理のずれが挙げられる．Xen 内部で直接 vNIC 制御を行なうなどして，内部処理のずれを縮小可能だと考えられる．

vNIC 内バッファについては，同期保存・復元対象外の計算機から大量にパケットが送信された場合，バッファが溢れる可能性がある．これは，現在無差別に受信したパケットを保存するためである．系間の整合性合わせに必要なパケットのみを保存するなどの，フィルタリングを組み入れることで，バッファ溢れについては回避可能だと考えられる．

保存・復元対象外の計算機の例として，クライアントの存在が挙げられる．VM 内のサーバアプリケーションに対し通信を行っているクライアントから見た場合，保存・復元の前後においてサーバ側の情報が不定となり，クライアント側の動作に支障を来す．これを解決するには，クライアント側も同期保存・復元の対象とする必要がある．

9. おわりに

本稿では，VM の同期保存・復元手法について述べた．

本手法では，複数の VM を同時に保存・復元する際に VM と vNIC バックエンドを切り離すことで，通信停止が生じないよう vNIC ばで通信を保存・復元し，保存時にネットワーク内に滞留するパケットの欠落を防ぐ．また，デバイスマッパーを用いることで，複数のディスク状態の保存，ならびに，その保存状態からのスナップショット作成を行う．これにより，クラスタミドルウェアが動作している複数の VM の，同期保存・復元を実現する．

評価の結果，従来，障害試験においてクラスタミドルウェアの系間状態の修復に 20 分要していたものを，VM の同期復元では 1 分程度で完了することを確認した．

今後の課題としては，より短い周期で通信を行うクラスタミドルウェアへの対応，保存パケットの整理，クライアント間との整合性考慮が挙げられる．

参 考 文 献

1) 庄内　亨：IT システム高信化技術の動向と課題-メインフレームからオープン時代へ-，電子情報通信学会 信学技報， Vol.107, No.17, pp.25–30 (2007).
2) 海老原寛之：システム開発における仮想化技術の活用，日本ソフトウェア産業協会論文集， Vol.9, No.09010, pp.1–25 (2009).
3) kernel.org: The Linux Kernel Homepage (2011). http://www.kernel.org.
4) Daniel P.Bovet, M.C.: 詳解 Linux カーネル 第 3 版 (2007). 高橋 浩和, 杉田 由美子, 清水 正明, 高杉 昌督, 平松 雅巳, 安井隆宏　訳.
5) Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A.: Xen and the art of virtualization, *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, New York, NY, USA, ACM, pp.164–177 (2003).
6) Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I. and Warfield, A.: Xen and the art of virtualization, *SIGOPS Oper. Syst. Rev.*, Vol.37, pp.164–177 (2003).
7) RedHat: Device-mapper Resource Page (2011). http://sources.redhat.com/dm/.
8) RedHat: LVM Resource Page (2011). http://sources.redhat.com/lvm/.
9) RedHat: LVM2 Resource Page (2011). http://sourceware.org/lvm2/.
10) RedHat: Dmsetup - low level logical volume management. Man Page. dmsetup(8).