IaaS クラウドにおけるスナップショットを用いた障害対策手法の提案

2011se061 市川 大誉　2011SE274 土本 大貴
指導教員：宮澤 元

1　はじめに

クラウドコンピューティング (クラウド) と呼ばれるインターネットを介してサービスを提供するコンピュータの利用方法が普及している．クラウドは複数のサーバを集約することで，効率的にコンピューティングリソースを活用出来るとともに，冗長性を利用して耐障害性を高めることが出来る．クラウドの種類は，アプリケーションを提供する SaaS(Software as a Service)，ミドルウェアを提供する PaaS(Platform as a Service)，ハードウェアや基盤ソフトウェアを提供する IaaS(Infrastructure as a Service) に大別される．

SaaS，PaaS では, 多数の物理ホストを利用して，ファイルを複製するなどして，耐障害性を高めている．一方，IaaS クラウドでは，ユーザはコンピューティングリソースを論理的なサーバである仮想マシン (Virtual Machine) の形で利用する．仮想マシンは，単一の物理マシンに依存して動作しているので，クラウドの冗長性を利用して耐障害性を高めるのが困難である．

IaaS クラウドにおいて障害復帰を容易にするために様々な工夫がなされている．例えば, 物理サーバの障害により，VM が停止した場合には別の物理サーバで VM を再起動するフェイルオーバーが提供されている．しかし，フェイルオーバーを行うと VM を再起動するので，障害発生時点の状態は失われる．

我々は，VM の状態を保存するスナップショットの機能に着目し，これを障害対策に利用すべく検討を進めている．スナップショットとはある時点で VM のディスクまたは，メモリの状態を保存することが出来る仕組みである．ユーザが必要な時に必要な物理サーバ上で復元出来るが，通常，処理のタイミングはユーザに任される．

本稿では，IaaS クラウドの障害対策手法としてスナップショット法を提案する．スナップショット法とは，スナップショットを繰り返し取得し，物理ホストの障害が発生した場合，状態が失われるのを防ぐために，スナップショットを利用して復帰させる障害対策手法である．スナップショット法の実現に向けて，様々な状況での VM の状態をスナップショットで取得し，最適なスナップショット取得方法について検討する．また，繰り返しスナップショットを取得する障害対策手法の実装を行う．

2　IaaS クラウドの障害対策

IaaS クラウドはクラウド基盤ソフトウェア上に構築されており，障害対策もクラウド基盤ソフトウェアによって提供されていることが多い．

クラウド基盤ソフトウェアは，様々な種類がある．Amazon 社の商用サービスとして Amazon EC2[1] や，OSS(Open Source Software) のプロジェクトである CloudStack[2] や OpenStack[3] などがある．本節ではクラウド基盤ソフトウェアが持つ様々な障害対策について述べる．

2.1　フェイルオーバー

VM が動作している物理ホストが故障した場合，別の物理ホストで VM を再起動させる機能をフェイルオーバーという．これを自動化して行なう自動フェイルオーバーも一般的に利用されている．フェイルオーバーでは，ファイルやデータベースに保存された情報は失われないが，それら以外の VM の動作状態などは失われてしまう．商用クラウドでは，Microsoft Azure[4] の Azure Site Recovery[5] や Veeam の Veeam Backup & Replication[6] などのように高機能化されたフェイルオーバーが提供されている．

図 1 は，フェイルオーバーを使った障害復帰を表している．横軸を物理ホスト上における VM の処理の流れ，’開始’ は VM の動作開始を表す．フェイルオーバーは，上の物理ホストで障害発生してしまうと，下の別の物理ホストで再起動して，一からやり直すことになる．しかし，フェイルオーバーでは通常動作中のオーバーヘッドはない．

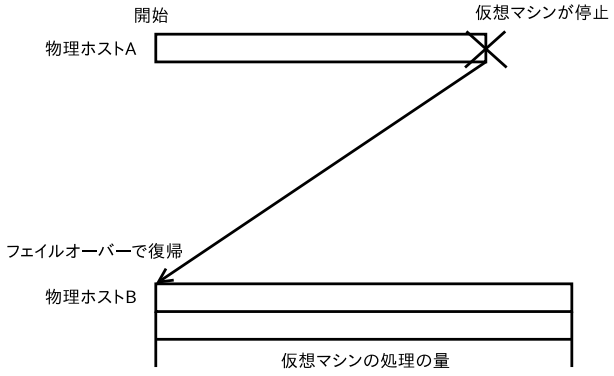

図 1　フェイルオーバーを利用した障害復帰

2.2　マイグレーション

VM が動作している物理ホストのメンテナンスや部品の交換が必要な場合，マイグレーションで別のホストに VM を移動させることが出来る．メモリとディスクの内容を引き継ぐことが出来るので，VM 上のサービスを移動先でもそのまま継続出来る．マイグレーション時に一旦 VM を止めなければならない通常のマイグレーション機能の他

に，VM を動作させたまま別のホストに移動させるライブマイグレーションがある [8]．VM の状態を失うことなく VM の動作を継続できるが，マイグレーション前にマイグレーション元の物理ホストが故障するような場合には利用できない．

2.3 スナップショット

スナップショットとは，VM 動作中のある時点でメモリやディスクを含めた VM の状態を取得し，保存する機能である．VM が動作している物理ホストが故障した場合，別のホストでスナップショット取得時の状態から再開できる．

スナップショットを復元することでフェイルオーバーのように再起動する必要はなくなるが，直前に取得したスナップショットまでしか戻れないので，それ以降の VM の状態は失われる．また，スナップショットを取りすぎるとデータの転送量が増え，ディスクの容量を圧迫してしまう．これにより，VM のテンプレートや VM のファイルシステムが保存できなくなるなど，新たに障害が増える可能性がある．

図 2 にスナップショットを用いた障害復帰の例を示す．障害が発生してもスナップショットを取得した時点まで VM の状態を戻すことが出来る．これを利用することで，VM を再起動して状態が失われるよりもデータの内容は更新されているので，保守性は高い．

一方，スナップショットを利用するには通常動作中にオーバーヘッドがかかる問題がある．

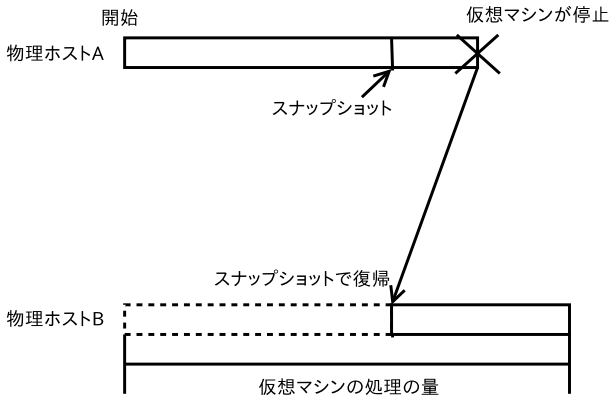

図 2 スナップショットを利用した障害復帰

2.4 VM レプリケーション

Remus[9] では，障害復帰のダウンタイムを最小限に抑えるために，VM レプリケーションを用いた障害対策を提案している．VM の状態を常にバックアップ VM に反映させることで，障害発生時にも VM の状態を失わず即時復帰が可能である．しかし，バックアップ VM のためにリソースを二重に用いる必要がある他，VM の状態を常にバックアップ VM に反映することによって，性能に大きな影響を与えることが避けられない．

3 スナップショット法

我々は，スナップショットを利用した障害対策として，スナップショット法を提案する．**図 3** に示すように短い時間間隔で繰り返しスナップショットを取り，障害発生時に，最新のスナップショットを用いて復帰することにより，障害により失われる状態を最小限に留める．さらに，スナップショットを取った時点ならどこからでも復帰することも可能になる．

スナップショット法では，スナップショットの頻度を高めれば，スナップショットで障害発生時に近い状態に戻せるが，スナップショット取得のオーバーヘッドが問題となる可能性がある．また，VM と外部との間の通信を考慮すると，単純にスナップショットを復元するだけで，障害発生時の状態を完全に復元するのは困難であり，スナップショットからの復元後に，復元不可能な通信をリセットするなどの処理が必要になると考えられる．

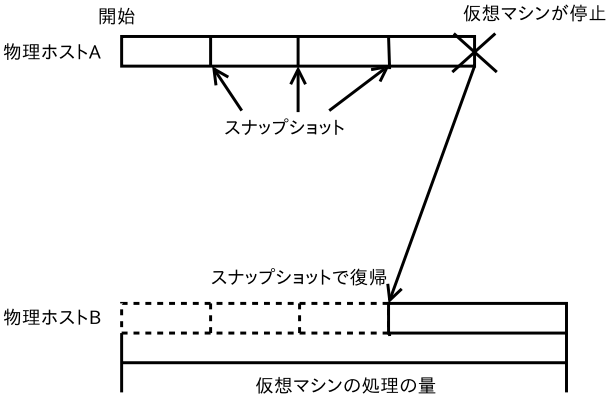

図 3 スナップショット法を利用した障害復帰

4 実験

スナップショット法実現の可能性を検討するために，実験を行った．

4.1 実験環境

IaaS クラウドを想定した実験をするため研究室内に，CloudStack を構築した．ハイパーバイザーとして，KVM をインストールする．CloudStack の構築環境を**表 1** に示す．また，実験で作成する VM の環境を**表 2** に示す．実験によって VM のメモリ量を変化させ，他は共通の環境にした．

4.2 VM のメモリ量とスナップショット取得・復帰時間

VM のメモリ量を変えてスナップショットの取得と復帰の時間を測定するための実験を行った．VM のメモリがクリーンな状態の際に最適化されることを避けるために，VM 上でメモリを確保し，その領域に書き込みを行い続ける負荷プログラムを動作させ，測定を行った．各メモリ量ごとに，負荷プログラムが確保するメモリ量を変えて，それぞれ，5 回ずつ測定し，平均を求める．

表 1　実験に用いた環境

OS	Ubuntu 64bit server 14.04
CPU	Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz
コア数	4(スレッド数 8)
HDD	821GB
メモリ	16.3GB
台数	4
ネットワーク	1000Base-T Ethernet LAN

表 2　VM の環境

OS	CentOS 5.5
CPU	QEMU Virtual CPU version 2.6.0
コア数	1
HDD	20GB
メモリ	1GB ～ 5GB

4.2.1　スナップショット取得時間

VM のメモリ量と平均スナップショット取得時間の関係を**図 4** に示す．負荷プログラムがメモリを確保していない状態では，VM のメモリ量にかかわらずスナップショット取得時間はほぼ一定である．

また，負荷プログラムが確保したメモリが VM のメモリ量以下ならばスナップショット取得時間は VM のメモリ量に比例して増加する．

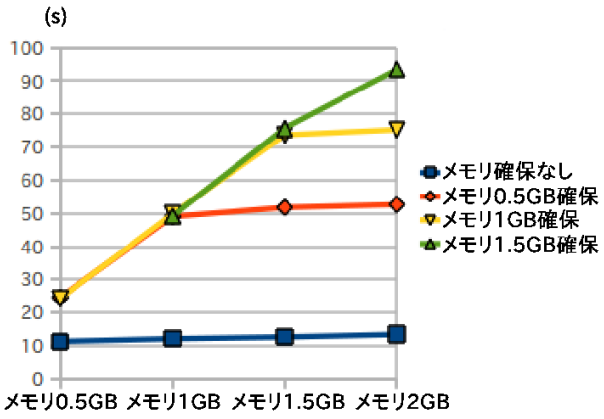

図 4　平均スナップショット取得時間

4.2.2　スナップショット復帰時間

VM のメモリ量と平均スナップショット復帰時間の関係を**図 5** に示す．スナップショット取得時間と同様に負荷プログラムがメモリを確保していない状態では，VM のメモリ量にかかわらず，スナップショット復帰時間はほぼ一定であることが言える．

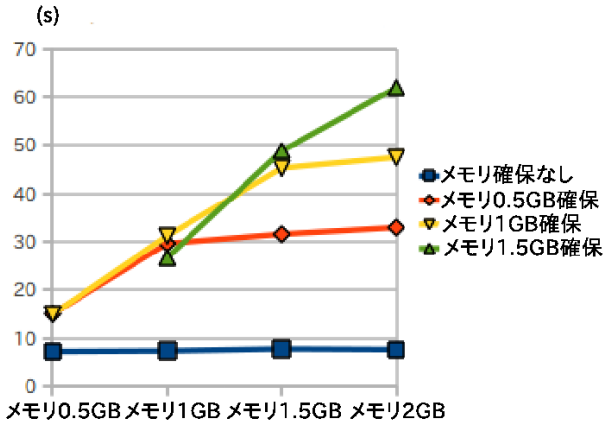

図 5　平均スナップショット復帰時間

4.2.3　フェイルオーバーとの比較

比較のためにフェイルオーバーにかかる時間も測定した(**表 3**)．フェイルオーバーの時間はメモリ量にかかわらず，ほとんど変化しない．

表 3　フェイルオーバーの時間 (s)

メモリ	時間 (s)
1GB	48.29
2GB	47.89
3GB	48.45
4GB	48.76
5GB	48.6

表 3 と**図 5** を比較すると，スナップショットを用いた場合，負荷プログラムがメモリを確保していない状態では，フェイルオーバーよりも早く復帰できることがわかる．

4.2.4　考察

フェイルオーバーの時間と負荷プログラムがメモリ確保していない状態でのスナップショットの取得と復帰を合わせて比べて見ると，フェイルオーバーよりも早く障害復帰することが出来るので，VM 起動時にスナップショットを取っておくことでフェイルオーバーより迅速に復帰出来るケースがあることがわかった．

一方，負荷プログラムがメモリを確保した状態では，特にメモリ確保量が多い場合にフェイルオーバーよりも時間がかかる場合が多い．

5　議論

前節の実験結果とその考察に基づき，様々な検討を行った．

5.1　スナップショット法の試作

負荷プログラムのメモリ確保量が多い場合，スナップショットの取得や復帰に時間がかかる．この場合，スナッ

プショットの取得頻度が高すぎると，システムの性能に悪影響を与える可能性がある．そこで，スナップショットの取得を低負荷の時のみ行うために，CPU 使用率を考慮してスナップショットを繰り返し取得するシステムを実装した．以下にシステムの動作の流れを示す (**図 6**).

1. VM の CPU 使用率の情報を取得する．
2. 1 の CPU 使用率が低い場合，スナップショットの取得間隔を短く，高い場合長く設定する．
3. 2 で設定した時間，待つ．
4. スナップショットを取得する．
5. 1 に戻る．

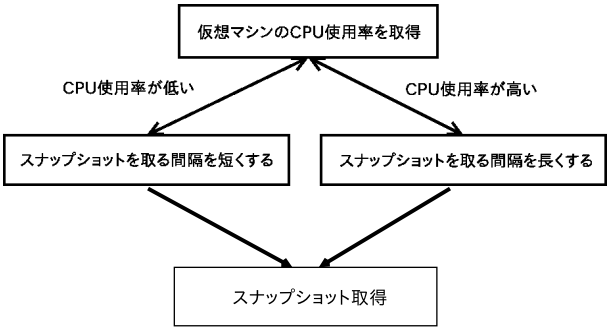

図 6 システムの動作の流れ

5.2 差分スナップショット法

VM の使用メモリ量が多い場合に，スナップショットの取得・復帰時間が長くなる問題を軽減するために，スナップショットの取得時に前回との差分だけを取得する方法が考えられる．これを差分スナップショット法と呼ぶことにする．差分スナップショット法の実現可能性を検討するために，物理ホスト上で，様々なアプリケーションのメモリ使用量を測定した．具体的には，proc ファイルシステムを利用してアプリケーションを一時的に動作させた状態で書き込みが行われたメモリ量を取得し，1 秒ごとの平均値を求めた．結果を表 4 に示す．

表 4 単位時間当たりのメモリ使用量

アプリケーション	メモリ使用量 [KB/s]
Firefox	31.5
QEMU	326

CPU 使用率が高い場合は，メモリ使用量も増えると考えられるが，VM の全メモリ容量から**表 4** のようにごくわずかなメモリ容量を取得すればよいとすれば，差分スナップショット法では，スナップショット取得，復帰にかかる時間が格段に減り，負荷を抑えることができる可能性がある．

しかし，差分スナップショット法の実現には以前取ったスナップショットとの差分をどのように取るか，差分を用いた VM イメージの作成をどのようにするかなど技術的な課題が多い．

6 まとめ

我々は，IaaS クラウドの障害対策としてスナップショット法を提案し，その実現可能性を検討するためにスナップショット取得時間など必要なデータを調べた．また，実験結果に基づき，CPU 使用率に応じて，スナップショットを取る頻度を変えるシステムを試作した．このシステムでは，CPU 使用率が高いときはスナップショットを取る間隔が長くなるので，状態の保存という目的では適していない．そこで，今後は差分スナップショット法について検討を進める予定である．

参考文献

[1] Amazon EC2, http://aws.amazon.com/ec2,(access 2014-10-8)

[2] openstack CLOUD SOFTWARE, http://www.openstack.org/,(access 2014-10-8)

[3] Apache CloudStack:Open Source Cloud Computing, https://cloudstack.apache.org,(access 2014-10-6)

[4] Azure: Microsoft **のクラウド プラットフォーム**, http://azure.microsoft.com/ja-jp/,(access 2014-12-26)

[5] Azure Site Recovery - Microsoft http://azure.microsoft.com/ja-jp/services/site-recovery/,(access 2014-12-26)

[6] Veeam **バックアップ、レプリケーション、リストア** - Veeam Backup \& Replication, http://www.veeam.com/jp/vm-backup-recovery-replication-software.html,(access 2014-1-6)

[7] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt and Andrew Warfield, “Live Migration of Virtual Machines,” in Proceedings of the Second USENIX Symposium on Networked Systems Design and Implementation (NSDI’05), pp. 273–286, 2005.

[8] Brendan Cully, Geoffrey Lefebvre, Dutch Meyer, Mike Feeley, Norm Hutchinson and Andrew Warfield, “Remus ：High Availability via Asynchronous Virtual Machine Replication,” in Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation(NSDI’08), pp.161–174, 2008.