2008－OS－108 (1)
2008／4／23

仮想機械を利用したサーバ間の協調的バックアップ基盤

大和﨑 啓 † 杉木 章義 ‡ 加藤 和彦 ‡†

† 筑波大学大学院 システム情報工学研究科
‡ 科学技術振興機構 CREST
E-mail: {k-yamato, sugiki}@osss.cs.tsukuba.ac.jp, kato@cs.tsukuba.ac.jp

要 旨

近年，インターネットサービスに高い可用性が求められている．一般的に可用性を高める手法として複数の物理計算機に複製する手法が広く知られている．しかし，この手法は複数の物理計算機を用意する必要がありコストが高く，特に小さなサービスで広域的に複製を行う場合，利用が難しい．本論文では，仮想機械を利用したPeer-to-Peer型の協調バックアップ基盤を提案する．サービスを提供する物理計算機同士が互いにサービスの複製を行うことで，少ない計算機資源で耐障害性をサービスに持たせている．また，仮想機械を利用することで，サービス間の独立性を高め，アプリケーションに改変を加えることなく複製や移送を行うことができる．本機構のプロトタイプを仮想機械にXenを利用し，当研究室で開発が進めているサステーナブルサービス基盤の一部として実装および実験を行った．その結果，障害発生時に複製サーバがサービスを再開し，サービスに耐障害性を持たせられることを確認した．

A Virtual Machine-based Cooperative Platform for Server Backups

Kei Yamatozaki† Akiyoshi Sugiki‡ Kazuhiko Kato‡†

† Department of Computer Science, University of Tsukuba
‡ CREST, Japan Science and Technology Agency
E-mail: {k-yamato, sugiki}@osss.cs.tsukuba.ac.jp, kato@cs.tsukuba.ac.jp

Abstract

Service availability is crucial for today's Internet services. Traditionally, replicating service state on multiple physical hosts is widely used for high availability. But, preparing multiple machines in geographically diverse environment is expensive, especially for small Internet services. In this paper, we present a virtual machine-based cooperative platform for service backup. The advantage of our platform is twofold: First, by enabling many Internet services to back up their services cooperatively, the platform provides high availability with minimum cost. Second, by using virtual machines, the platform encapsulates the complete states of the services. Furthermore services can share a physical host while maintaining a high level of isolation. We have implemented a prototype of our platform and evaluated it by experiments. The results of our experiments show that participating hosts can back up each others, and that services can recover from host failures.

1 はじめに

近年，インターネット上で多くのサービスが提供され，利用者も増加している．インターネットサービスの重要性が増すにつれ，障害によるサービスの停止による損害も大きくなり [6]，高い可用性を持った高信頼なサービスが求められている．

一般的に可用性を高める手法として，複数の計算機にサーバのデータを複製する手法が広く用いられている．しかし，この手法は物理計算機を複数台用意するため，高いコストを支払わなければならず，特に小さいサービスにおいて敷居が高い．複製を行う計算機はサービスごとに必要となるため，小さなサービスにおいてはサービスの大きさに対して運用コストが高く，採用したくてもできないのが現状である．また，ディザスタ・リカバリを目指して広域分散環境で複製を行おうとすると敷居はさらに高くなり，一部のサービスに導入が限られてしまう．

本論文では，仮想機械を利用した Peer-to-Peer 型の協調バックアップ基盤を提案する．それぞれの計算機が互いにサービスを複製し合うことで，少ない資源で高い可用性を持った複数のサービスを提供することができる．サービス提供サーバが他のサービスの複製サーバとしての役割を兼任することもでき，障害発生時には，複製サーバが最新のスナップショットからサービスを稼働させ，障害が発生した計算機に代わってサービスの提供を開始する．

本機構では，さらに仮想機械を導入し，導入のしやすさとサービスの信頼性を高めている．サービス複製に仮想機械を利用する利点は 4 つある．まず，アプリケーションを改変する事なくサービスアプリケーションの複製が行える．次に，独立性により仮想機械同士が互いに影響を与えることなく，計算機資源を共有することができる．3 つめは OS，ライブラリなど必要な環境全てをカプセル化しているため，サービスの移送を容易に行うことができる．最後に，動作状態からの回復である．仮想機械はプログラム実行中の動的なメモリ，CPU などの状態を保存することができ，ディスクのイメージから再起動することなく回復できる．

当研究室では，サステーナブルサービス基盤 [13] の開発を進めている．サステーナブルサービス基盤は，広域分散環境において多数の計算機が連合し，システム上で運用しているサービスの可用性と信頼性を高めることを目指したシステムである．本研究はサステーナブルサービス基盤の応用例として設計を行い，分散環境でのサーバ管理と高信頼なサービス複製を実現した．また，本論文は [10] の論文の発展版であり，改良のための議論や広域分散環境に向けた設計について示す．

本機構を実装し，実験を行った．仮想機械には仮想マシンモニタの Xen[1] を利用した．実験では，各物理計算機の CPU ロードアベレージを測定しサービスの代替動作の確認，サービスのスループットを測定することでクライアント側からのサービスの動作確認を行った．その結果，本機構を利用したサービスに障害を発生させても，正しく複製サーバがサービスを代替えし，サービスの継続が可能であることを確認した．

2 サービス複製における仮想機械の利点

従来のディザスタ・リカバリではストレージの複製のみを行い，サービスはストレージのデータから復旧させる場合が多い．サーバの複製を行う場合は，同一 LAN 内の HA クラスタであることが多く，同一ディスクを共有しながら切り替える．これらのシステムは特殊なハードウェアやアプリケーションを必要とし，高い可用性を実現している反面，コストが高い．

仮想機械をサービス複製に用いることは以下の 4 つの利点がある

- **透過性**: サービスアプリケーションの複製を行うためには，アプリケーションの改変が必要となることが多い．仮想機械ではアプリケーションを改変する事なくサービスの複製を行える．改変によるコストの削減と共に特定のサービスに依存しない高信頼化を実現できる．
- **独立性**: 仮想機械は実ハードウェアに近い独立性を提供している．そのため，セキュリティや障害，サービスの問題などが同一計算機内の他のサービスに影響を与えることがなくなり．本機構では，サービス提供サーバが複製サーバとしての役割も兼任することがあり，独立性による安全な計算機資源の共有が重要となる．
- **完全性**: サービスを複製する場合，計算機環

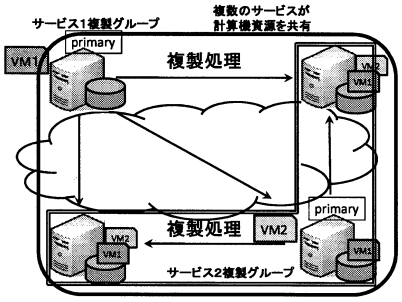

図 1: 協調的バックアップ

境による依存関係の問題が生じることがある．仮想機械では OS やライブラリと言ったサービスに必要な環境全てをカプセル化しているため，依存関係を意識することなく移送を行える．また，広域分散環境を想定した場合においても同様のことが実現でき，従来の複製手法に比べ広域分散への対応が容易となる．

- **動作状態からの回復**: 仮想機械はプログラムのメモリ，CPU をはじめとする実行状態などの動的な状態を保存することができる．そして従来のストレージを利用した静的な回復手段に比べて短時間で回復できる可能性がある．

3 仮想機械を利用した協調的バックアップ

本論文では，仮想機械を利用したサーバ間の協調的バックアップ基盤を提案する．図 1 のように複数のサービスを計算機同士が協調して，プライマリサーバと複製サーバでサービス複製グループを構成し，提供を行なう．サービス複製グループ同士は計算機資源を共有するため，サービスごとに必要となるコストを低く抑えることができる．また，仮想機械を用いることでサービスの複製によって生じる独立性の問題を解決する．さらに，本機構は自由な参加や離脱を容易にするため，中央管理ノードの存在しない非集中型の協調バックアップ基盤である．

3.1 要件

仮想機械の協調バックアップを実現するには 4 つの満たすべき要件がある．

- **サービス複製グループの構成**: サーバの Peer-to-Peer 実現するためには各サービスで独立した複製グループを管理しなければならない．本機構は非集中型であるため，各グループが独立に判断を行なっても協調して動作し続ける管理手法が必要である．

- **障害発生時のグループ再計算**: 障害が発生することでサービス提供サーバや複製サーバが失われてしまい，可用性の低下に繋がってしまう．これを回避するため，複製を行なうグループの再計算が必要である．

- **物理計算機のメンバ管理**: 本機構では，動的に計算機が参加・離脱することができる．そのため，これらの動作による物理計算機の変化にも対応できる必要がある．この物理計算機からサービスごとに選択し複製を行なう．

- **サービスの高信頼な複製**: 仮想機械を利用したサービス複製では複製の配布や障害発生時の回復を適切に行なう必要がある．

3.2 サービスの複製グループの構成

本機構では，各サービスのプライマリが複製を行う複製グループを選択し，サービスごとに独立した管理を行う．物理計算機群の各計算機にハッシュ関数として SHA-1 を使用し，物理計算機とサービスにユニークなハッシュ値を与え，管理を行っている．

ハッシュ値を元に，物理計算機群で仮想的なリングを形成し，Chord[9] 型のオーバーレイを構成する．各計算機はそれぞれ独立して計算機リストを持っているが，各計算機で独立に計算しても同一のリストを構成することができる．サービス提供サーバはサービスのハッシュ値に最も近い値を持つ計算機が選ばれる．そして，サービスのハッシュ値を中心として前後の計算機にサービス複製の配置を行う．このとき，サービスのハッシュ値にサービス提供を行わせたい計算機と同一のハッシュ値を与えることによって，サービス提供サーバを指定している．

複製グループの構成例を図 2(a) に示す．この物理計算機群ではサービス 1，2 を提供し，サービス 1 はホスト B で提供し，ホスト A，C に複製を配置している．同様にサービス 2 をホスト C で提供し，ホスト B，D に複製を配置している．そして，ホス

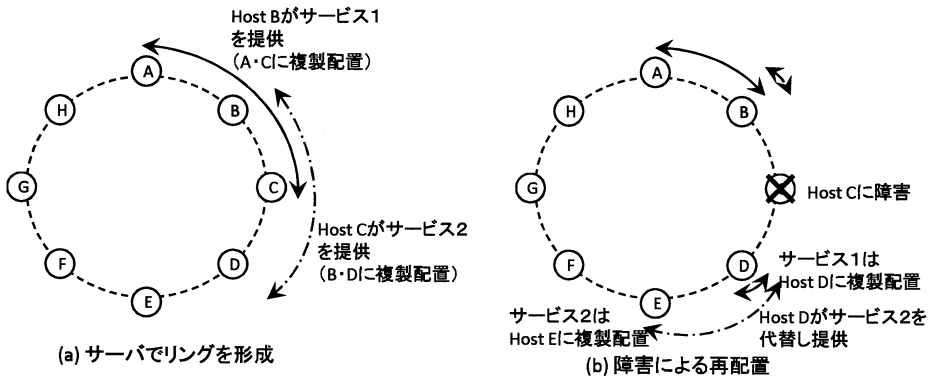

図 2: サービスの配置例

トB，A，Cをサービス1，ホストC，B，Dをサービス2の複製グループとしている．

このようにして，非集中型のシステムであっても，各計算機が同一の判断を下し，協調して動作を行う．

3.3 障害からの復帰と複製の配置

障害が発生することで，複製サーバまたはサービス提供サーバが失われることがある．複製サーバが失われてた場合，障害が発生した計算機を除いた物理計算機群から，新たなサーバを選出し，複製グループを再構成する．サービス提供サーバが失われた場合，複製グループの中から相応しいサーバが最新のスナップショットからサービスを起動させ，サービスの代替を行う．

図2の(b)では，ホストCに障害が発生した場合である．これを受け，サービス1は複製サーバであったホストCに代わりホストDを複製サーバにしている．サービス2はサービス提供サーバが失われてしまったため，ホストDがサービス2を代替し，さらにホストEを複製サーバとして補充している．結果，障害によりサービス1はホストB，A，Dが，サービス2はホストD，B，Eが複製グループとなる．

このような動作をすることで，障害が発生しても複製サーバの台数が減ってしまうことなく，サービスの可用性が低下してしまうことを防いでいる．

3.4 物理計算機群のメンバ管理

システムに参加している全ての計算機の管理をメンバシップ機構で行なう．計算機の参加・離脱は動的に行われ，全ての参加計算機は参加しているすべての計算機を知ることができる．そのため，それぞれのサービスは任意の計算機をバックアップに選ぶことができる．現在，このメンバシップ機構はスケーラビリティと堅牢性を持つGossipプロトコルによって実現している．Gossipプロトコルでは自分の持っている計算機のリストをランダムに他の計算機に広めることを行う。

3.5 サービスの高信頼な複製

仮想機械上のサービスの複製には，合意アルゴリズムの一種であるPaxosを用いて行なう．仮想機械の状態を他のサーバに複製する場合，複製の途中で障害が発生すると，必ずしも最新の状態から回復するとは限らない．例えば，半数のサーバのみに複製が完了し，障害が発生した場合，どのサーバから回復するかによって結果が異なってしまう．また，全てのサーバに対する複製の完了を待ちながら処理を進めると，遅いサーバの性能や一時的な離脱に全体が支配されてしまう．さらに，障害発生時に1台の計算機のみが正しく代替えすることを保証しなければならない．

これらの問題をPaxosを利用しサーバ間で合意をとることによって，最新のコミットされたスナップショットから，唯一の計算機が代替えすることを保証する．また，過半数の計算機から応答が得られ

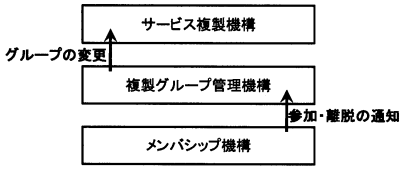

図 3: 協調的バックアップ

れば良いため，遅い計算機や一時的な障害の影響を避けることができる．プロトコルの詳細は [14] に示す．

3.6 全体の構成

以上から，全体の構成は図 3 のようになる．メンバシップ機構は物理計算機群の状態を複製グループ管理機構に通知する．その情報を基に複製グループを決定する．そしてサービス複製機構に複製グループの情報を通知することによって，複製を配置し複製グループの構成が完了する．障害発生時には，メンバシップ機構が障害を検知し，複製グループ管理機構に通知する．障害発生によって複製グループに変更がある場合，サービス複製機構に通知し，新たな複製グループの構成をする．

4 実装

本機構は当研究室で開発を進めているサステーナブルサービス基盤 [10] の一部として実装した．生産性を優先するため Java で実装されており，本機構も同様に Java で実装を行った．

本研究では，仮想化技術としてオープンソースの仮想マシンモニタである Xen[1] を利用している．なお，性能を優先させるため準仮想化を利用しており，近年のバージョンで実装されたチェックポイント機構を利用している．従来からも仮想機械の一時停止，再開は提供していたが，他の計算機へ移送し，動作させることができなかった．チェックポイント機構は他の計算機に移送し，再開することができる．

5 実験

本論文では，本機構によるオーバヘッドと障害発生時の振る舞いについて実験を行う．5.2 節では Xen のオーバーヘッドの測定を行い，5.3 節ではチェックポイントに要するコストの測定を行った．5.4 節では，障害発生時にサービスの代替することを確認する．5.5 節にクライアントから見たサービスのスループットの変化を示す．

表 1: Xen によるオーバヘッド

	Native[MB/s]	VM[MB/s]	ratio[%]
dbench	421.48	257.71	61.1
tbench	83.42	67.03	80.3

5.1 実験環境

実験には 4 台の CPU Xeon 3.60GHz，メモリ 2GB，SCSI 接続のディスクで構成された計算機を使用した．全ての計算機は 1000BASE–T で単一のスイッチに接続されている．ホスト OS には Fedora 8（Linux 2.6.21-2952）を使用した．Xen は 3.1 を使用し，ゲスト OS もホスト OS 同様 Fedora 8(Linux 2.6.21-2952) を使用し，256MB のメモリを割り当て，準仮想化で起動させた．Java は 1.5 を使用した．

ベンチマークには dbench スイート [11] の dbench と tbench コマンドを使用した．dbench は samba の任意の数のクライアントをシミュレートし，ファイルシステムの負荷を発生させてディスク I/O の性能を測定することができる．一方 tbench は，ネットワーク部分をエミュレートしネットワーク帯域を測定することができる．

5.2 Xen によるオーバヘッド

Xen のオーバヘッドの測定では，仮想機械とネイティブ環境で，dbench と tbench で 5 つのクライアントをシミュレートし I/O ベンチマークとネットワークベンチマークを行った．tbench による測定では同一 LAN 内にクライアントを用意して測定を行っている．

比較を行った結果を表 1 に示す．仮想機械は I/O ベンチマークではネイティブ環境の 61.1% と半分以上の性能が出ていた．ネットワークベンチマークでは 80.3% と高い性能が出ていた．これらの結果から，Xen による仮想化ではサービスを提供するうえで十分な性能が仮想機械で出ていると言える．

表 2: チェックポイントのコスト

	チェックポイント取得間隔 [s]		
	30	60	120
取得時間 [s]	8.45	8.42	8.42
可用性 [%]	78	87.6	93.4
復帰時間 [s]	8.08	8.09	8.1
サイズ [MB]	268.7	268.7	268.7

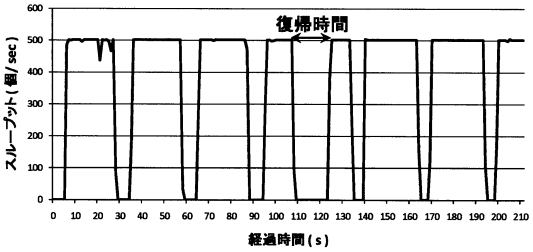

図 5: クライアント側でのスループットの測定

5.3 チェックポイントの取得コスト

チェックポイントの所要時間として，チェックポイントの取得時間とチェックポイントからの復帰時間を測定した．チェックポイントをとる際，仮想機械に負荷を与えるために dbench を実行させて測定し，30 秒，60 秒，120 秒とチェックポイント取得間隔を変えてそれぞれ 20 回ずつ取得した．また，チェックポイントからの復帰時間を測定した．

結果を表 2 に示す．フルチェックポイントのためチェックポイントの取得時間，復帰時間，チェックポイントのファイルサイズに違いは全くなかった．チェックポイント取得中は仮想機械が停止してしまうため，取得間隔から取得時間を引いた割合がサービスの可用性となる．サービスの負荷による取得時間に変化がないため，取得間隔が伸びれば，当然一定時間あたりの可用性は向上している．しかし，チェックポイントの間隔を延ばすと，障害発生時にサービスの状態がより以前の状態へ戻るため，間隔を長くすることはあまり好ましくない．間隔を 30 秒に設定した状態の可用性は，80% 未満と低い数値になっている．実際にサービス提供を行う上でこの問題は大きい．そのため，実際のサービス運用にはチェックポイントの取得時間の短縮が課題となる．

5.4 サービスの代替動作実験

次に，本機構上で動作するサービスが正しく回復することを確認する．4 台のホストマシン A〜D を用意し，サービス 1，2 を稼働させ，CPU ロードアベレージの測定を行った．仮想機械上で動作させるサービスは，CPU 負荷を与えるため無限ループを行なうプログラムである．一定時間後にサービスのプライマリに障害を発生させ，それぞれの計算機の CPU ロードアベレージの変化を測定した．

結果は図 4 の通りである．まず，サービス 1 のプライマリはホスト A でホスト B，C がバックアップとなっている．サービス 2 のプライマリはホスト D で，ホスト B，C がバックアップとなっている．スナップショット取得と配布は 30 秒ごとに行っており，それに伴うホスト OS の定期的なロードアベレージの変動が各ホストで確認できる．スナップショット取得時には，仮想機械の動作を停止させるため，サービス VM によるロードアベレージが一時的に 0 になっていることがわかる．100 秒後にホスト A にて障害を発生させている．障害を検知すると，すぐに回復動作が行われ，サービス 1 がホスト B にて代替されていることが確認できる．同時に，サービス 1 の複製サーバを新たに 1 台補充するため，ホスト D がサービス 1 のバックアップとなる．この補充動作によるロードアベレージの大きな変動がホスト A の障害発生とほぼ同時期に起こっていることが確認できる．180 秒後にホスト A の復帰によりサービス 1 のバックアップが C と A になっている．そして 240 秒後にホスト D にて障害が発生し，ホスト C がサービス 2 のプライマリとなり，ホスト A，B がバックアップとなっている．

これらの結果から，サービスに障害が発生したことによる代替動作が行われ，サービスが正しく行われていることが確認できる．

5.5 クライアントから見た変化

同一 LAN 内にクライアントを用意し，サービスへ UDP パケットを送受信するサービスを動作させる．そしてクライアント側でスループットの記録を行う．サービスの稼働を確認した後，サービス提供サーバを停止させ，別のサーバでサービスが立ち上がる動作がクライアント側でどのように見えるのかを調べた．同時に，サービス復帰までの時間も測定

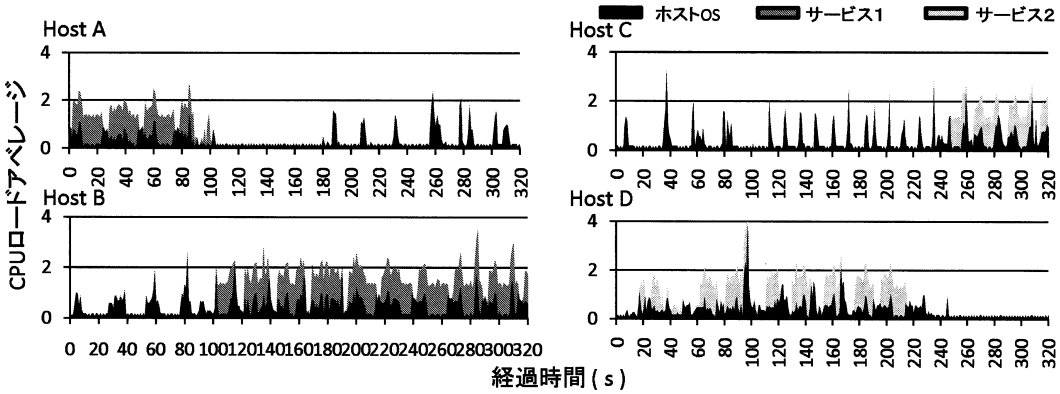

図 4: 各ホストの CPU ロードアベレージの変化

した.

結果は図 5 のようになった. まず, 定期的なスループットの低下が確認できる. これは, チェックポイントの取得時に仮想機械が停止し, サービスの提供も停止するためである. 約 8 秒の仮想機械の停止がクライアント側から確認できる. グラフの 107 秒付近で障害を発生させた. それによって, クライアント側ではサービスが停止していることが確認でき, その後にスループットが回復していることも確認できる. サーバに障害が発生してから, 他のサーバが障害を起こしたサーバに代わってサービスの提供を開始するまで約 16 秒間で行われていることがわかる.

従来のサービスでは, サーバに障害が発生し停止してしまうと, 障害の原因解明や複製サーバを使ってデータを回復する必要があったが, この実験結果から, クライアント側からは多少のスループット低下は確認できるものの, すぐにサービスが再開されているため, サーバの障害をあまり意識することなくサービスの利用が行えることがわかる.

6 関連研究

仮想機械を基にした耐障害性やディザスタ・リカバリを目指した研究が行われている. これらの研究は, 仮想機械自身を拡張することを目的としており, 我々の目的としている様々なアプリケーションに耐障害性を実現させる協調的バックアップ基盤の開発とは異なる.

広域分散環境を想定し, 仮想機械を利用したディザスタ・リカバリ機構として Second Site[3] がある. Second Site は Xen の live migration 機能を改変し, 仮想機械内のメモリ操作やディスク操作など, イベント単位で複製を行っている. VM-FIT[7] は仮想機械のアクティブ複製により耐障害性を実現する研究である. また, Stodden らの研究では Xen のセミ・アクティブ複製 [8] をおこなっている. これらの研究は仮想機械の複製手法のみを扱っており, 協調基盤を目指す我々の研究とは異なる.

協調基盤の研究として, ストレージシステムに主眼をおいた Glacier[5] や Cooperative Proxies[12], その他の Peer-to-Peer システム [4] がある. 我々の研究では仮想機械と Peer-to-Peer を結合し, コスト効率が高く, 高可用性を与える協調基盤をインターネットサービスに提供することを目指している.

7 広域分散環境への対応

本論文では, 協調バックアップ基盤の中心機構について示したが, 実際に広域分散環境上で利用するためにはいくつか解決しなければいけない課題がある.

まず, 障害発生時に別拠点の仮想機械にリクエスト先を切り替えるアドレス切り替えの問題が発生する. アドレス切り替えは LAN 環境であれば, gratuitous ARP で容易に実現することが可能であるが, 広域分散環境上では拠点同士が異なるネットワークとなるため難しい. IP 層で対応する方法, DNS を利用する方法, クライアント側で実装する方法などさまざまな解決策があるが, それぞれにトレードオフの関係がある. 本論文では実現の容易さ

から，Dynamic DNS を利用したアドレス切り替えを目指していく．この手法は，DNS サーバを用意するだけでよく，Akamai や Google でも利用されている手法である．仮想機械の IP アドレスを DNS エントリとして登録しておき，TTL を 5 分程度と短く設定しておく．障害が発生すると，対応するエントリの IP アドレスを代替先の IP アドレスに置き換える．セキュリティの問題から仮想機械内に鍵を保存しておき，該当する仮想機械のみが対応する DNS エントリ書き換え可能とする．この手法は，代替先への切り替えに数分程度要するが，実現が容易でアプリケーションも書き換える必要がない．

次に，チェックポイントのネットワーク転送の問題である．本機構は複製プロトコルを利用することで遅いノードの影響や一時障害を避けているものの，フル・チェックポイントを利用しているため転送量自体が大きい．Remus[2] などではすでに copy-on-write チェックポイントを実現しており，これらの手法と本論文の協調基盤を組み合わせることでサービスの可用性をより高めることができる．

8 まとめと今後の予定

本論文では，仮想機械を利用したサーバ間の協調的なバックアップ基盤を提案した．本機構は少ない計算機資源で，サービスに高い可用性を持たせることができる．また，仮想機械を利用することで，アプリケーションに改変を加える必要なく複製を行ない，同一計算機を共有した場合にも独立性を保つことができる．

本機構をサステーナブルサービス基盤上で実装し，評価を行った．仮想機械には Xen を利用した．実験を行ったところ，障害から正しく復帰し，サービスに耐障害性を持たせることができた．

今後，広域分散環境への対応を行い，評価を行なう.. また，チェックポイントの速度や複製方式の改良を行う．最後に，計算機の CPU 使用率など，計算機資源を考慮したサービス複製サーバの選定を行う．

謝辞

本研究は科学技術振興機構 CREST「自律連合型基盤システムの構築」の支援を受けている．

参考文献

[1] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. *ACM SOSP'03*, pp. 164–177, 2003.

[2] B. Cully, G. Lefebvre, D. Meyer, M. Feeley, and N. Hutchinson. Remus: High availability via asynchronous virtual machine replication. *USENIX NSDI'08*, 2008(To appear).

[3] Warfield A. Cully, B. Secondsite: Disaster protection for the common server. *USENIX HotDep'06*, 2006.

[4] S. Elnikety, M. Lillibridge, M. Burrows, and W. Zwaenepoel. Cooperative backup system. *USENIX FAST'02 WiPs*, 2002.

[5] A. Haeberlen, A. Mislove, and P. Druschel. Glacier: Highly durable, decentralized storage despite massive correlated failures. *USENIX NSDI'05*, pp. 143–158, 2005.

[6] D. Patterson, A. Brown, P. Broadwell, G. Candea, M. Chen, J. Cutler, P. Enriquez, A. Fox, E. Kiciman, M. Merzbacher, D. Oppenheimer, N. Sastry, W. Tetzlaff, J. Traupman, N. Treuhaft. Recovery oriented computing (ROC): Motivation, definition, techniques, and case studies. *UC Berkeley CSTR*, Vol. UCB//CSD-02-1175, , 2002.

[7] H.P. Reiser and R. Kapitza. Hypervisor-based efficient proactive recovery. *IEEE SRDS'07*, pp. 83–92, 2007.

[8] D. Stodden. Semi-active workload replication and live migration with paravirtual machines. *Xen Summit, Spring 2007*, 2007.

[9] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM'01*, pp. 149–160, 2001.

[10] A. Sugiki, K. Yamatozaki, and K. Potter, R.and Kato. A platform for cooperative server backups based on virtual machines. *International Service Availability Symp.'08*, 2008(To appear).

[11] A. Tridgell. dbench. `http://samba.org/ftp/triage/dbench/`.

[12] A. Wolman, M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy. On the scale and performance of cooperative web proxy caching. *ACM SOSP'99*, pp. 16–31, 1999.

[13] 小磯和之, 阿部洋丈, 鈴木与範, Richard Potter, 池嶋俊, 加藤和彦. サステーナブルサービスための基盤ツールキットの設計. IPSJ Trans: コンピューティングシステム, Vol. 48, No. SIG 3(ACS 17), pp. 13–26, 2007.

[14] 杉木章義, Richard Potter, 加藤和彦. 仮想機械のスナップショット機構を利用したサービスの高信頼なパッシブ複製手法. 情報処理学会研究報告 (2008-OS-107), pp. 79–86, 2008.