情報処理学会論文誌　コンピューティングシステム　Vol. 2　No. 1　1–11 (Mar. 2009)

広域分散環境のための仮想機械を利用したサービス協調複製基盤

杉木章義[†1]　大和﨑啓[†2]　加藤和彦[†1,†2]

近年のインターネットサービスは厳しい時間と資源制約のもとに開発されている．サービスに対する高い可用性が求められているものの，短い開発期間からサービスに複製処理を追加し，十分なテストを行うことは難しい．また，離れた拠点に複製を行う広域複製が潜在的には望まれているものの，多くの場合，コストの問題から導入を諦めているのが現状である．本論文では，広域分散環境における仮想機械を利用したサービス協調複製基盤を提案する．まず，サービス提供者がお互いに計算機資源を提供し，サービスを複製しあうことで少ない参加コストでサービスの可用性を高めることができる．次に，仮想機械を用いることでサービスごとの独立性を高め，特定のサービスに依存しない複製処理や再開機能を提供することができる．本基盤を Xen と Scrap Book for User-Mode Linux の 2 つの仮想機械とともに実装し，日本国内の 3 拠点で実験を行った．その結果，2 つのサービスを同時に稼働させた場合でも，障害から回復することが確認された．

A Cooperative Virtual Machine-based Service Replication Platform for WAN Environments

Akiyoshi Sugiki,[†1] Kei Yamatozaki[†2]
and Kazuhiko Kato[†1,†2]

Recent Internet services have been developed at extremely limited time and resources. Although service availability is highly demanded, it is generally difficult to achieve high availability within short developing time. Remote replication is a potentially demanded solution to protect a service from whole site down, but it is hardly adopted from small service providers due to unacceptable costs. In this paper, we present a cooperative virtual machine-based replication platform for wide-area environments. We provide a cost-effective, highly-available solution via service state replication over peer-to-peer platform. Virtual machines increase reliability over such platform by isolating each other and capturing complete service state. We have implemented a prototype supporting two virtual machines, Xen and Scrap Book for User-Mode Linux. Our experimental results show that the prototype successfully replicated two services on three sites in Japan and recovered them in case of failures.

1. はじめに

近年，インターネットサービスでは可用性がサービスの重要な側面の 1 つとなっている．伝統的に可用性を高める手法として，複数の計算機にサービスの状態を複製する手法が用いられ，ディザスタリカバリ[1] や拠点内のフォールトトレラントシステム[2] として実現されている．これらの方式は非常に高い信頼性を実現し，企業システムを中心に広く利用されている反面，インターネットサービス，特に単一計算機で提供される小規模なサービスでは厳しい開発期間制約やコスト制約からそれほど利用されていないのが現状である．

インターネットの性質の 1 つであるロングテール性により，単一計算機で提供されるサービスはインターネットのいたるところで見ることができる．これらのサービスには，小中学校などの教育機関，非営利団体，ベンチャー企業などのウェブサーバやメールサーバなどが含まれ，Google，Amazon などの少数の巨大サービスが存在するのと対照的に規模が小さいながら数多く点在している．これらの単一計算機で提供されるサービスでは，可用性を向上させるために複数の計算機を購入し，管理することは現実的に難しい．特に，地理的に離れた拠点に複製を行う広域複製は管理コストが大きく，潜在的には望まれながらほとんど利用されていない．そのため，サービスの可用性が限定されたものとなっている．もし，複数の計算機を購入し，複数の拠点に配置できたとしても，多くの資源が利用されないままとなり，無駄が発生する．

本論文では，仮想機械を利用したサービス協調複製基盤を提案する．ここでいう協調とは，サービスがお互いの状態を複製しあうことであり，単一計算機で提供される小規模なサービスが他の計算機上に状態を複製することで，少ない参加コストで高い可用性を持ったサービスを実現することができる．さらに，仮想機械を利用することで協調基盤の信頼性を高めている．仮想機械はアプリケーション，ライブラリ，オペレーティングシステム（OS）などサービスの完全な状態をカプセル化し，複製することができる．しかも，従来の多くの

†1 科学技術振興機構 CREST
CREST, Japan Science and Technology Agency
†2 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba

耐障害システムのようにアプリケーションやライブラリを改変することなく，特定のサービスによらない複製方式を実現できる可能性がある．また，仮想機械の独立性により，同じ物理計算機を複数のサービスで共有してもお互いの影響が及びにくい．

本基盤は仮想機械をパッシブ待機方式で複製する．パッシブ待機とは，仮想機械のスナップショットを複製する方式である．本基盤は高信頼マルチキャストを利用しており，複数の物理計算機にスナップショットが正しく複製されていることを保証する．障害が発生した場合，整合性のとれた最新のスナップショットから再開が行われる．物理計算機はメンバシップによって自動的に分散管理され，任意のタイミングで協調基盤に参加や離脱することができる．

本基盤を Xen[3] と Scrap Book for User-Mode Linux（SBUML）[4] の 2 つの仮想機械とともに実装し，日本国内の 3 拠点を利用し，実験を行った．その結果，2 つのサービスを同時に稼働させた場合にも，障害から回復することが確認された．

本論文は，文献 5) の発展版であり，広域分散環境のための機能を追加し，広域分散環境で実験を行っている．また，小磯らの研究[6] は単一サービスを想定したツールキットであったのに対して，本基盤は複数サービスに対応した協調基盤であり，複製方式や障害が発生した場合の振舞いも異なっている．

本論文の構成は以下のとおりである．2 章では，サービス協調複製基盤の概要について説明する．3 章ではシステム構成の詳細について記述する．4 章で実装を示し，5 章で実験の概要と結果について記述する．6 章で関連研究との比較を行い，7 章で本論文をまとめる．

2. サービス協調複製基盤

本章では，サービス協調複製基盤の概要を示す．

2.1 想定環境とサービスモデル

図 1 に本基盤の概要を示す．現在の基盤は数台から数十台の物理計算機で動作しているが，将来的には数千から数万台規模を目指している．また，本基盤には**計算機資源提供者**，**サービス提供者**，サービスの**クライアント**の 3 種類が参加している．

まず，本基盤では計算機資源提供者が物理計算機の提供を行う．本基盤は広域分散環境で動作するため，それぞれの計算機が独立したディスクを持つ．これは共有ディスクを仮定し，LAN 内で動作する VMware HA[7] や Bressoud と Schneider の研究[8] などの仮想機械の HA クラスタの仮定とは異なる．また，本基盤は現在，簡単のためそれぞれの計算機を信頼している．信頼できない環境での動作は今後の課題である．本基盤は統一した資源ビューを提供し，物理計算機は任意のタイミングで参加したり離脱することができる．

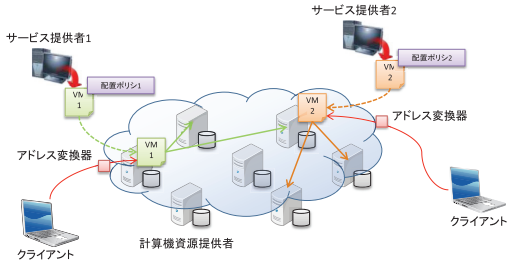


図 1　サービス協調複製基盤
Fig. 1　A cooperative service replication platform.

次に，サービス提供者はサービスを仮想機械でカプセル化し，任意の計算機からインストールすることができる．現在，サービスとしては，電子メールサーバやウェブサーバなどの単一計算機で提供できる比較的軽量なインターネットサービスを仮定している．サービス提供者が与えた**配置ポリシ**に従って，仮想機械が自動的に複数の物理計算機に配置され，複製処理が開始される．仮想機械に障害が発生すると，再びポリシに従い，他の物理計算機でサービスが再開される．このポリシはサービスごとにサービス提供者が設定することができ，サービスどうしのポリシの競合によって実際の配置が決定される．

最後に，サービスのクライアントは本基盤が提供する**アドレス変換器**を通じて，仮想機械内のサービスにアクセスする．仮想機械に障害が発生すると，アドレス変換器が変更を反映し，クライアントは代替を行った物理計算機上の仮想機械にアクセスすることができる．

2.2 仮想機械を用いる利点

仮想機械を利用したアプローチは特定のサービスによらない複製手法を実現できる可能性があり，主に実現の容易さに利点がある．このほかの特徴も含めて，主なものを次で説明する．

- **透過性：**仮想機械を利用することでサービスの状態を外部から透過的に取得することができる．
- **完全性：**仮想機械はアプリケーション，ライブラリ，OS など，完全な状態をカプセル化することができる．
- **独立性：**仮想機械は物理計算機に近いセキュリティを提供することができる．また，仮想機械内で障害が発生しても，他の仮想機械に障害が及ばない．
- **資源効率：**同一計算機上で複数の仮想機械を動作させることができる．また，近年の

サーバは十分なディスク容量を持っているため，多数のスナップショットを蓄積しておくことができる．

- **実行状態からの回復：**データベースを利用した複製では，回復時に代替先でシステムの起動処理が必要となるが，仮想機械ではプログラムの実行状態ごと保存し，復元することができる．

以上から，仮想機械の利用は魅力的であり，本論文では仮想機械を利用した方式を目指す．

2.3　仮想機械の複製方式

従来から，サービスの途中状態を複製する方式として**状態機械方式**（state machine approach）が広く知られている．ここでいうサービスの状態には，サーバにあるクライアントのデータやサービス全体で共有するデータなどが含まれる．この手法はクライアントから 1 台の計算機として見える一方で，実際には複数の計算機が連続的に同じ状態へと遷移している．この手法は従来からのレガシーなサービスをそのまま複製対応にできる一方で，スケーラビリティが 1 台のままに制限される．

この手法は仮想機械と相性が良く，特定のサービスによらない複製方式が実現できる可能性がある．複数の仮想機械が同じ状態へと次々遷移すれば，複製が実現できる．

状態機械方式には大きく分けて，アクティブ待機とパッシブ待機の 2 種類がある[9]．それぞれの特徴を**表 1** にまとめる．なお，これらの派生としてセミアクティブ待機，セミパッシブ待機などがあるが，基本的にどちらかを元にしている．

- **アクティブ待機：**この方式ではクライアントからサーバに対するリクエストを複製し，複数の計算機に入力する．そして，独立にリクエスト処理を行い，同じ状態へと遷移する．この手法は各計算機が独立に処理を行うため計算コストが高いが，通信コストは抑えられるという利点がある．

  アクティブ待機は広く利用されている反面，それぞれの計算機が割込みや CPU スケジューリング，時計などのイベントによらず，同一の入力から同じ状態へと遷移することを仮定する．これをサービスが**決定的**（determinstic）であると呼ぶ．
- **パッシブ待機：**この方式では，**プライマリ計算機**と呼ばれる 1 台の計算機でクライアントからのリクエスト処理を行い，その実行後の状態（スナップショット）を複数の**バックアップ計算機**に転送する．各計算機の状態をプライマリ計算機で一意に定めるため，サービスは決定的でなくてもよく，バックアップ計算機の計算コストも小さい．その反面，クライアントからのリクエストに比べ，スナップショットの方が一般的に大きいため，通信コストが大きい．

表 1　複製方式の比較
Table 1　Comparison of active and passive standby.

	複製方式	サービス	CPU コスト	ネットワークコスト	段階的な回復
アクティブ待機	リクエストを複製	決定性に限る	×	√	×
パッシブ待機	スナップショットを複製	非決定性でもよい	√	×	√

本論文ではパッシブ待機の手法を用いる．本手法は特定のサービスに依存しない手法を目指すため，サービス状態が一意に定まる決定性を仮定するのは難しい．また，バックアップ計算機の計算コストが小さいため，サービスが複数あった場合にそれらのバックアップ計算機をまとめ，少数の計算機で運用することができる．さらに，最新のスナップショットから回復できなかった場合，過去のスナップショットをたどり，段階的に回復を試行することもできる．

近年，仮想機械でサーバの状態複製を行う研究が行われ始めているが，これらは本手法とは異なる．VM-FIT[10] はアクティブ待機を行っており，Stodden[11] は Xen でセミアクティブ待機を行っている．Remus[12] は Xen のパッシブ待機に近いが，協調基盤を作成する本論文の目的とは異なる．

3. システム構成

図 2 に各計算機のシステム構成を示す．本ソフトウェアはホスト OS 上で動作し，すべての計算機に同一のソフトウェアがインストールされている．本ソフトウェアはメンバシップ層，サービス配置層，複製プロトコル層の 3 階層で構成しており，以後，下層から上層に向かって順に説明する．

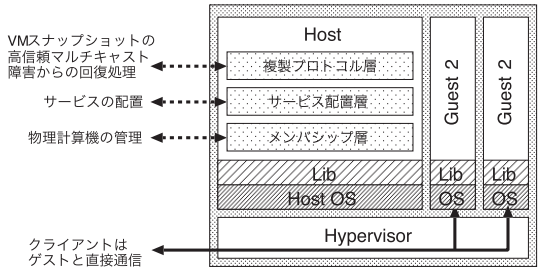


図 2　システム構成
Fig. 2　System architecture.

- **メンバシップ層：**メンバシップ層は物理計算機の参加や離脱を管理する．統一的な資源ビューを提供し，障害の検出はこの階層で行われる．
- **サービス配置層：**サービス配置層はメンバシップ層が作成した資源ビューをもとに仮想機械と物理計算機のマッピングを管理する．
- **複製プロトコル層：**複製プロトコル層はサービス配置層で決定された配置をもとに，高信頼プロトコルを使用し，仮想機械の状態を複製する．障害が発生した場合，自動的に代替が行われる．

また，このほかにも仮想機械のアドレス割当てや障害発生時の切替え機構も保持している．以下，3.1 節から 3.3 節にかけて各階層の詳細について説明し，3.4 節でアドレス切替え方式を説明する．

3.1　メンバシップ層

すべての参加している物理計算機はメンバシップによって自動的に管理される．メンバシップはすべての計算機に統一された資源ビューを提供し，構成が変化するたびにサービス配置層にイベントを通知する．

資源ビューは $\{< h_1 : s_1, s_2 >, < h_2 : s_2, s_3 >, < h_3 : null >, \ldots\}$ などの形式で提供される．h_1，h_2，h_3 は物理計算機の IP アドレスを表し，s_1，s_2，s_3 は物理計算機で動作している仮想機械を表す．仮想機械は SHA-1 ハッシュ値で管理される．

現在，本基盤のメンバシップは階層型 Gossip プロトコル[13] をもとにしている．本論文では，物理計算機単位でなく，その上で動作する仮想機械単位で管理できるよう修正している．Gossip プロトコルでは，定期的に自身が知っている資源ビューをランダムに選択した他の計算機に送信する．受け取った計算機は自身の資源ビューと比較し，反映する．この動作を繰り返すことで，メンバシップが全体に行きわたる．すべての項目にはタイムスタンプが付加されており，一定時間更新が行われないと障害が発生したものとして資源ビューから取り除かれる．さらに本論文では階層型を使用しており，近隣の計算機であるほど頻繁に更新されるようにしてスケーラビリティを高めている．

また，本基盤は追加の障害検出機構を自由に導入することができる．そのため，独自の障害検出機構を実装すれば，さまざまな障害に対応することができる．

3.2　サービス配置層

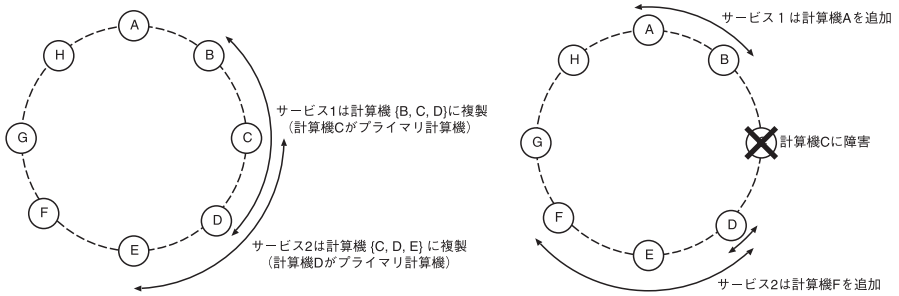
サービス配置層では，仮想機械とその複製の配置を決める．本基盤は分散管理されているため，サービスごとにサービス提供者が独自の配置ポリシを設定することができる．

実際の配置はサービスごとに存在するプライマリ計算機が決定する．一方，すべての物理計算機にはローカルな計算機資源を管理する**資源ブローカ**が存在し，自身の計算機に仮想機械をインストールしてよいかどうか許可を与える．プライマリ計算機はポリシに従いながら，メンバシップ層の情報をもとに，それぞれの計算機の資源ブローカと交渉し，許可が与えられると実際に配置する．また，グローバルな資源量の調整もプライマリ計算機が行う．

ポリシは Java のクラスに対応するプログラムで与え，物理計算機の参加や離脱などのイベントごとに記述する．また，任意のタイミングで強制的に配置を変更することができる．これを利用して，固定台数の計算機につねに複製しておくポリシや，物理計算機が追加されるたびに貪欲に複製の追加を行うポリシなどを記述することができる．なお，複数のサービスでポリシが競合する可能性があるが，その解決手法は個別のポリシに任されている．同一物理計算機上で複数の仮想機械間で性能保証を行う場合など，より高度な資源管理手法については今後の課題である．

すべてのサービスが同じポリシに従い，どの物理計算機で配置を計算しても同じ結果となるならば，全体として統一された動作を行うこともできる．本論文では，この方式のポリシとして consistent hashing 型[14] の配置を提供している．まず，すべての参加している物理計算機から SHA-1 ハッシュ値を計算し，リング状の空間上に物理計算機を配置する．サービスも SHA-1 ハッシュ値によって管理されていることから，サービスのハッシュ値に最も近いハッシュ値を持つ物理計算機にプライマリ計算機を配置する．バックアップ計算機は空間上のプライマリ計算機の前後の計算機に配置する．

図 3 に動的な構成変化の例を示す．最初，サービス 1 は物理計算機 {B, C, D} で動作し，サービス 2 は物理計算機 {C, D, E} で動作している．ホスト C，D はそれぞれのサービス



(a) 物理計算機 C の障害発生前　　(b) 物理計算機 C の障害発生後

図 3　動的な計算機構成変化の例
Fig. 3　Example of dynamic host configuration.

のプライマリ計算機である．ホスト C に障害が発生すると，サービス 1 は物理計算機 {A, B, D} に変化し，サービス 2 は物理計算機 {D, E, F} に変化する．

3.3　複製プロトコル層

複製プロトコル層は，仮想機械の安全な複製と回復を保証する．本論文の手法は，Paxos [15),16)] をもとにした高信頼マルチキャストを使用し，パッシブ待機方式で複製を行う．Paxos はよく知られたプロトコルであり，安全性を保証しながら，遅い計算機や一時的な障害を避け，非同期的に複製を行う．本論文では，Mazières のハンドアウト[17)] をもとに，仮想機械の複製のために最適化を行っている．

本手法は次の事柄について仮定する．

- 本論文では，物理計算機の Fail-Stop 障害のみを仮定し，ネットワークは信頼できるとする．しかし，この仮定は Byzantine 障害に対応した状態複製手法[18)] など，最新の成果を導入することにより解決することができる．
- 物理計算機 F 台の障害に対処するため，$2F+1$ 台の物理計算機を必要とする．しかし，この仮定は最も基本的な Paxos を使用しているためであり，改良版[19)] を使用することにより，必要な台数を削減することができる．
- 本論文では，サービスのリクエスト処理とは独立に非同期的に複製を行うため，最新の整合性のとれたスナップショットから回復できることのみを保証する．一般的に，サービスの処理性能と障害発生時の回復地点の間にはトレードオフの関係があり，広域分散環境で同期的に複製を行ったのではサービスの処理性能が著しく低下してしまう．そのため，本論文ではサービスの処理性能を優先し，複製にかかる時間に応じて複製間隔を調整する．

図 4 に本論文の複製プロトコルを示す．元となる Paxos アルゴリズムではプロトコル上にさまざまな役割が存在するが，本論文ではプライマリ計算機とバックアップ計算機の 2 つの役割に集約する．プロトコルには 2 つの状態があり，障害が発生していない場合の (a) **通常状態**と障害が発生した場合の (b) **ビュー変更状態**の 2 つで構成される．

図 4 (a) に通常状態のプロトコルを示す．通常状態ではサービスの複製処理を行う．複製処理は仮想機械の完全スナップショットがすでに配布された状態で開始され，よりサイズの小さい差分スナップショットのみを継続的に複製し，完全スナップショットに反映していくことで最新の状態へと遷移していく．

基本的に複製処理は 1 往復のメッセージ通信で行う．ただし，一部の処理が次の送信時に便乗して行われる．まず，(1) プライマリ計算機で差分スナップショットを作成し，(2) バックアップ計算機に複製する．(3) バックアップ計算機は応答を返し，(4) プライマリ計算機は自身を含め過半数以上の計算機から応答を受け取ると，複製が安全に完了したものとして差分スナップショットを完全スナップショットに反映する．この時点で，バックアップ計算機上では差分スナップショットは完全スナップショットに反映されていないが，次の最初のメッセージ (1') を受信した際に，差分スナップショットを完全スナップショットに反映させる．このプロトコルの重なりにより，メッセージの往復が 1 往復となり，効率的に複製が行われる．また，過半数以上の計算機から応答が得られればよいため，遅い計算機や一時的な離脱による影響を避けることができる．一方で，一部の計算機が遅れることになるが，これらの計算機は次で説明するビュー変更プロトコルを利用することで最新状態に一致させることができる．

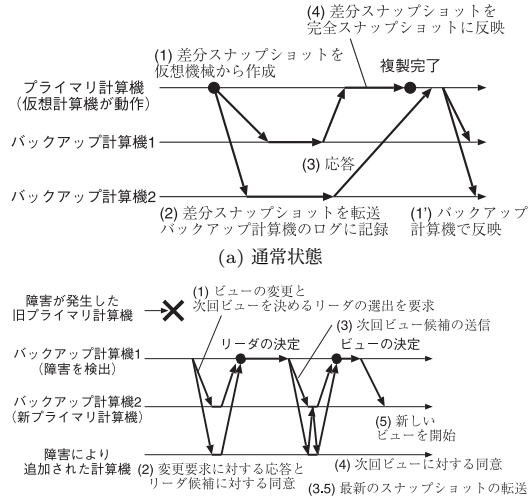


図 4　仮想機械の複製プロトコル
Fig. 4　Replication protocol for virtual machines.

障害が発生すると，ビュー変更状態に移行する．図 4 (b) にプライマリ計算機に障害が発生した場合のビュー変更プロトコルを示す．なお，バックアップ計算機に障害が発生した場合も同一のプロトコルを使用し，以前と同じ計算機をプライマリ計算機として選出する．ビュー変更状態では，プライマリ計算機を選出するとともに，最新の整合性のとれたスナッ

プショットを各計算機に配布する．

ビュー変更は基本的に 2 往復のメッセージ送受信で行う．最初の往復でビューの変更を要求し，次回の構成を決めるリーダを選出する．図 4 (b) の場合，投票によりバックアップ計算機 1 をリーダとして選出している．次の往復でプライマリ計算機とどの時点のスナップショットから再開させるかを決定する．これらは投票により決定し，もし最新の整合性のとれたスナップショットを持っていない計算機があれば，図 4 (b) の (3.5) のように途中で他の計算機から受け取るため，最新状態に追いつくことができる．図 4 (b) の場合，リーダのバックアップ計算機 1 がバックアップ計算機 2 を次回のプライマリ計算機と決定し，実行を再開させている．

3.4　仮想機械のアドレス管理

仮想機械のアドレス割当てとして，ブリッジ方式や Network Address Translation（NAT）方式が利用されている．本論文では，グローバル IP の消費を抑える目的から，宛先 NAT を利用し，物理計算機の IP の特定のポートに対するアクセスのみ宛先 IP アドレスを仮想機械の IP アドレスに書き換えている．また，ブリッジ方式にも少量のコード追加で容易に対応することができる．

ある仮想機械に障害が発生すると，アドレス変換器によってクライアントからのリクエストを新しい代替先へと切り替える必要がある．LAN 環境であれば，Gratuitous ARP [20] により容易に実現することができるが，広域分散環境では，クライアント側に実装する方法，プロキシを設置する方法，物理計算機側で実装する方法などさまざまな手法があり，相互に利点と欠点がある．

本論文では，Dynamic DNS [21] を利用したアドレス変換器を作成する．この手法は切替えに時間を要するが，多くのサービスで改変が必要ないという利点がある．また，この手法は広域ロードバランサで広く利用されている．障害が発生すると，代替先の計算機が DNS サーバに接続し，対応する DNS エントリを更新する．現在，本基盤の DNS エントリの TTL は一般的に受け入れられている最も短い時間に合わせて 120 秒としている．このほかにも，Onoue らの Quasar [22] や Bradford ら [23] のようにトンネルを併用する手法も試行しているが，元の物理計算機自身に障害が発生した場合には利用することができない．

4. 実　　装

本基盤のプロトタイプを SF2 と名付け，実装した．生産性を優先するため Java で記述しており，23,193 行で構成されている．本基盤は Java で実装されているが，高速化のために工夫を行っている．本基盤は内部を処理ごとにスレッドに分割し，イベント駆動で処理を行う SEDA [24] のような構成となっている．また，通常のメッセージ送受信には Java のシリアライズを利用するが，仮想機械のスナップショットなど大きなファイルの転送には sendfile に対応する I/O 機能で転送する．

また，本基盤は少量のコードの追加でさまざまな仮想機械とともに利用することができる．現在，Xen と SBUML の 2 つをサポートしている．

Xen は広く利用されているオープンソースの仮想機械である．Xen は対応したカーネルを必要とする反面，高速に動作することが知られている．最近の Xen では標準的に完全スナップショット機構を提供しているものの，ディスクに関しては状態を取得しない．そのため，本基盤では `blktap` を利用し，ディスクの差分スナップショットの実装を行った．これは Remus [12] や Bradford ら [23] の論文などでも実装されている機構と同等の機構である．同期ディスク I/O を行う `sync` モジュールを改変し，前回のスナップショットからの差分をメモリに保持しておき，名前付きパイプを経由した指令によりスナップショットとして書き出す．また，スナップショットどうしを結合する機能の実装も行った．

SBUML は User-Mode Linux の派生版であり，オリジナルがサポートしていないスナップショット機能を提供している．SBUML は TUN/TAP デバイス [25] をサポートする Linux であればどの環境でも利用可能である．TUN/TAP デバイスは標準的な Linux ディストリビューションであれば組み込まれた状態で配布されており，多くの環境で利用できる．SBUML も Xen と同様に，CPU やメモリに関しては完全スナップショットを取得し，ディスクに関して差分スナップショットを取得することができる．なお，本基盤の複製プロトコル自身は差分スナップショットに対応しており，仮想機械で完全な差分スナップショットが実装されれば，すぐに利用することができる．

5. 実　　験

本論文では，複製プロトコル自身の処理性能，仮想機械によるオーバヘッド，全体を統合した場合の複製性能，動的な変化に対する対処などの観点から実験を行う．

5.1　実 験 環 境

実験には，筑波大学 4 台，産業技術総合研究所（秋葉原）1 台，豊橋技術科学大学 3 台の PC サーバを使用した．**表 2** に拠点間の Round-Trip Time（RTT）と 1 GB のファイルの転送時間を示す．この結果は，TCP 版の ping と sendfile を使用したファイル転送で取得した．

表 2　各拠点間のネットワーク性能
Table 2　Network performance between sites.

拠点間	RTT [ms]	1 GB の転送時間 [s]
筑波大（LAN 内）	0.02	25.7
筑波大–産総研	10.4	575.7
産総研–豊橋技科大	19.2	610.9
豊橋技科大–筑波大	15.7	126.6

各拠点のすべてのサーバは同一性能であり，デュアル Xeon 3.60 GHz の CPU，2 GB のメモリ，SCSI 接続 36 GB のディスクで構成されている．ホスト OS には Fedora 8 を使用し，Xen 3.2.1 をコンパイルして使用した．ゲスト OS には Xen では Linux 2.6.21.7-3.fc8xen を使用し，SBUML では CentOS 3.9（Linux 2.4.24-1ub-1sb）を使用した．ゲスト OS には Xen，SBUML ともにすべて 256 MB のメモリを割り当てた．Java は Sun JDK 1.6.0 を使用した．

実験のベンチマークは dbench 3.0.4 [26] スイートを使用した．dbench スイートは dbench コマンドと tbench コマンドで構成されており，dbench は samba の負荷を模倣するベンチマークである．書き込みが 90%に近い割合を占めるため，仮想機械にとって非常に負荷の高いベンチマークである．tbench は samba の TCP 通信部分のベンチマークである．なお，クライアント数は 4 に設定した．

5.2　複製プロトコルの性能

まず，仮想機械によらない複製プロトコル自身のオーバヘッドを調査するため，実験を行った．その結果を**表 3** に示す．これは，何も処理を行わない null リクエストを 3 台に複製した場合の平均時間を示しており，LAN 環境では筑波大学内の 3 台，WAN 環境では各拠点 1 台の計 3 台を使用した．

LAN 環境では平均 4.93 ms であり，文献 2) で C 言語で記述した場合の平均 2.98 ms と同じオーダに収まっている．また，ビューの変更にかかる時間は複雑なメッセージ交換のため，106.6 ms と複製の場合より増加している．LAN 環境と WAN 環境を比較すると，WAN 環境では表 2 の RTT に相当する時間が加わるため，LAN 環境よりも時間が増加している．複製処理では 1 往復分の時間が増加し，ビュー変更では 2 往復分の時間が増加する．

5.3　仮想機械によるオーバヘッド

次に仮想機械による性能オーバヘッドを**表 4** に示す．ここでは，ネイティブ計算機の場合と SBUML，Xen の場合のスループットを比較している．Xen に関しては 3 通りの `blktap` を使用しており，非同期 I/O を行う `aio`，同期 I/O を行う `sync`，さらに `sync` をもとに本論文でスナップショット機構を追加した `sf2` の 3 つである．

表 3　複製プロトコルの処理性能
Table 3　Performance of replication protocol.

環境	複製時間 [ms]	ビュー変更時間 [ms]
LAN	4.93	106.6
WAN	37.0	193.7

表 4　仮想機械による性能オーバヘッド
Table 4　Performance overhead incurred by VMs.

ベンチマーク	ネイティブ	SBUML	Xen `aio`	Xen `sync`	Xen `sf2`	[MB/s]
dbench	445.71	16.65 (3.73%)	248.68 (55.79%)	27.53 (6.18%)	35.50 (7.96%)	
tbench	64.05	9.26 (14.5%)	54.34 (84.8%)	54.27 (84.7%)	54.41 (84.5%)	

括弧内はネイティブ性能に対する比率を表す．

表 5　スナップショット機構のオーバヘッド
Table 5　Snapshot and restoring overhead.

	SBUML		Xen	
	標準	gzip	標準	gzip
サイズ [MB]	96.9	14.9	317.2	31.9
（CPU，メモリなど）	48.6	14.8	256.6	31.4
（ディスク）	48.3	0.2	60.6	0.5
取得時間 [sec]	2.86	7.34	7.79	21.85
実行再開時間 [sec]	2.30	2.40	6.98	9.87

表 4 の dbench の場合を見ると，Xen で `aio` を使用した場合，ネイティブの 55.79%のスループットと仮想機械の間で最も高いことが分かる．SBUML は 3.73%であり，Xen の `sync` は 6.18%と `aio` に比べ大幅に低下している．`sf2` は `sync` をもとに作成しているため，同程度の性能であるが，オリジナルの `sync` よりもわずかに向上しているのは 1 度使用した領域をメモリ上にキャッシュするためである．もし，スナップショット機構を `aio` をもとに作成すれば，性能が大幅に向上する可能性がある．一方，tbench を使用した場合，総じて Xen の方が SBUML に比べ性能が高く，`blktap` のモデルにほとんどよらない．

表 5 はスナップショット機構のオーバヘッドを取得した結果である．dbench で 60 秒ごとにスナップショットを取得した場合の平均サイズ，平均時間について示している．総じて，SBUML の方がユーザレベルでエミュレーションを行うため，サイズ，時間ともに小さい．

表 6　仮想機械の複製性能
Table 6　VM replication performance.

環境	SBUML [sec]	Xen [sec]
LAN	5.03	18.93
WAN	7.66	24.76

また，SBUML は標準で sparse ファイルを使用するうえに，gzip による圧縮もサポートしているため，こちらを使用するとさらにサイズを小さくすることができる．一方で，取得や再開時間は増加している．Xen は標準ではメモリにをそのままダンプするため，メモリ割当て量にスナップショットサイズが比例する．しかし，本論文で SBUML と同等の圧縮機能を追加すると，サイズを 317.2 MB から 31.9 MB に大幅に縮小することができた．しかし，この機能はまだ最適化していないため，多くの取得と再開時間を要する．圧縮を行った場合，どちらの仮想機械もサービスの静止時間は標準の場合のスナップショット取得時間とほぼ同じである．

5.4　仮想機械の広域複製

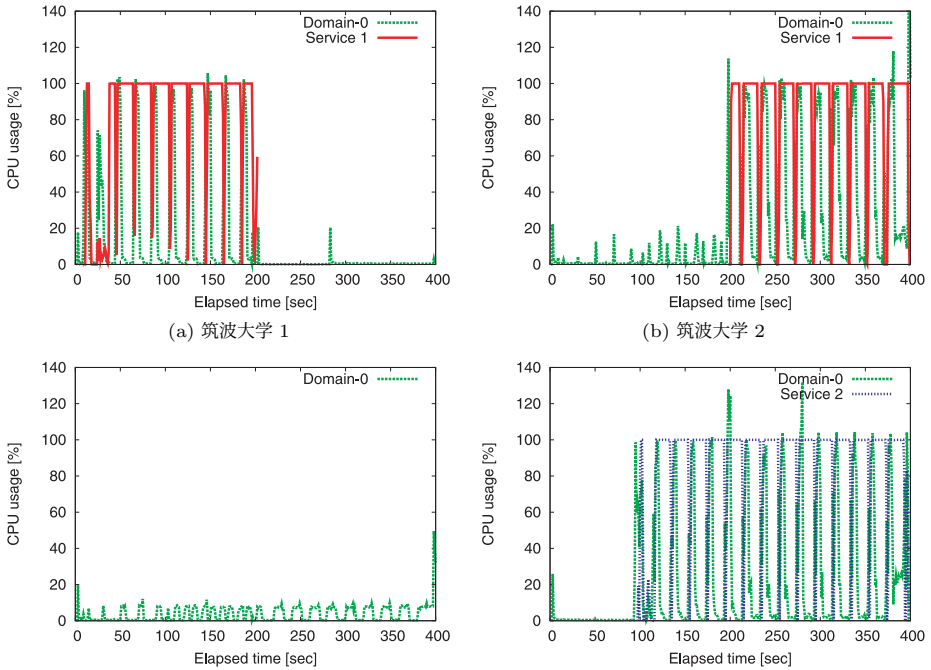
表 6 に Xen，SBUML で LAN 環境，WAN 環境の双方でアイドル状態の仮想機械を複製した場合の結果を示す．これは，表 5 に相当するスナップショットの取得を開始してから，過半数である 2 台に複製が終わるまでの時間を示している．どちらの仮想機械も gzip による圧縮を使用している．

どちらの仮想機械も転送時間の増加のため，WAN 環境の方が複製にかかる時間が増加している．また一方で，スナップショット取得時間が全体に占める割合が大きく，この時間の改善が必要である．アイドル状態という違いがあるものの，どちらの仮想機械も表 5 に相当するスナップショット取得時間がかかっている．Xen の live migration 機能を改造し，動作させながら高頻度に差分スナップショットを転送するシステムに Remus [12] などがある．また，仮想機械の転送の高速化はマイグレーションの文脈[27] でも行われている．これらの研究と本基盤を組み合わせることで 1 回あたりの複製時間を短縮し，複製間隔を大幅に縮めることができる．

また，仮想機械の完全なベースイメージを転送する場合，表 2 に相当する時間を要する．これらの時間が必要となるのは，仮想機械の初期インストール時や障害発生時に物理計算機を補充する場合に限られる．

5.5　動的な変化に対する振舞い

最後に，**図 5** に動的な計算機構成変化に対する振舞いを示す．仮想機械に Xen を使用し，



(a) 筑波大学 1　(b) 筑波大学 2
(c) 産業技術総合研究所（秋葉原）　(d) 豊橋技術科学大学

図 5　障害発生時の振舞い
Fig. 5　Handling dynamic changes.

筑波大学 2 台，産業技術総合研究所（秋葉原）1 台，豊橋技術科学大学 1 台の計 4 台の計算機を使用している．変化を際立たせるため無限ループで CPU 時間を消費する 2 つのサービスを実行し，xentop で CPU 使用率を 400 秒間監視した．なお，スナップショット間隔は 60 秒であり，配置ポリシはランダムに選んだ 3 台に複製するよう設定している．

まず，初期状態ではサービス 1 は筑波大学 1 で稼働しており，筑波大学 2 と産業技術総合研究所（秋葉原）の 2 台に複製されている．サービス 1 の CPU 使用率は 60 秒ごとに低下し，この期間はスナップショットを取得している期間である．サービス 1 の CPU 使用率回復直後，Domain 0 の CPU 使用率が毎回増加しているが，これはスナップショットの複製処理のためである．

100 秒後，豊橋技術科学大学の計算機にサービス 2 がインストールされ，筑波大学 2 と産

業技術総合研究所（秋葉原）の 2 台の計算機への複製処理が開始される．このようにサービスはいつでも追加または削除してよい．200 秒後，筑波大学 1 に意図的に障害を発生させている．本協調基盤はただちに障害を検出し，筑波大学 2 で代替処理が行われている．サービス 1 の複製を行っていた計算機が 1 台減少したため，豊橋技術科学大学の 1 台を加え，複製処理を継続している．280 秒前後で，筑波大学 1 が協調基盤に復帰し，この実験の後でさらに障害が発生すれば再び複製処理に利用される．

以上から，本基盤は動的な環境変化へと追従していくことができる．また，本システムは非集中型で分散管理されていることから，どの拠点の計算機に障害が発生しても全体として動作を続けることができる．

6. 関 連 研 究

広域分散環境を想定し，仮想機械を利用したディザスタ・リカバリを目指している研究として Second Site[28] がある．Second Site はポジションペーパのみが公開されており，まだ詳細が明らかにはなっていない．Remus[12] は Second Site から派生したプロジェクトであり，LAN 環境内で Xen を利用した複製機構を提案している．Remus は live migration 機能[29] を改変し，1 秒間に数十回程度の高い頻度で複製を行っている．本基盤でも Remus の成果を利用すれば，サービスの停止時間や複製間隔を大幅に改善することができる．また，Remus は複製プロトコルを使用しておらず，バックアップ計算機が複数あった場合に，どのように最新の状態からの回復を保証するか示されていない．さらには，本研究の協調基盤を作成する目的とは異なる．

そのほかにも，近年，仮想機械を利用した複製機構の研究が広く進められている．Stodden[11] は Xen でセミアクティブ複製を行っている．VM-FIT[10] では，Xen でアクティブ複製を行っている．アクティブ待機方式は転送コストが小さいという利点がある反面，決定性のサービスに限られる．Bressould らの研究[8] や近年の VMware HA[7] や田村ら[30] は LAN 内でディスクを共有する HA クラスタについて扱っている．これらの研究は非常に高い可用性を実現する反面，本研究はディスクを共有しない広域分散環境を仮定しており，これらの研究とは目的が異なる．

仮想機械を利用したサービス基盤の研究は広く進められ，広域分散環境を想定したものに Virtuoso[31]，vMatrix[32] などがある．Amazon EC2[33] でも最近になって地理的に離れた複数のデータセンタが利用可能となっている．Virtuoso は複数の仮想機械間でオーバレイを構成する手法を扱っており，vMatrix は Content Distibution Network を対象にしている．Amazon EC2 では仮想機械自身はステートレスであり，サービス状態はストレージである S3 に保存する．これらは，協調基盤上で複製処理を行う本研究とは異なっている．また，協調基盤の研究は Peer-to-Peer システムを中心に，Proxy[34] やストレージ[35] などのさまざまな文脈で進められている．本論文は仮想機械を利用した協調基盤の実現を目指している．

7. 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協調複製基盤を提案した．本基盤は広域分散環境を想定して作成されており，サービス提供者がお互いのサービスの状態を複製しあうことで少ない参加コストでサービスの可用性を高めることができる．また，仮想機械を用いることでサービスごとの独立性を高め，特定のサービスに依存しない複製処理や再開機能を提供している．本基盤を Xen と SBUML の 2 つの仮想機械とともに実装し，日本国内の 3 拠点で実験を行った．その結果，2 つのサービスで同時に複製処理を行うことができ，障害発生時に回復することが確認された．

今後は，本論文の成果をもとにさまざまな方向へ発展させていく．本論文では状態機械方式による可用性の向上を実現したが，同一基盤上で複数の仮想機械インスタンスを起動させることも可能である．今後は，スケーラビリティを重視した複製方式も実現していく．また，クライアントとの間にネットワークバッファを設置し，停止時間を隠蔽するなど，現在のモデルについても改良していく．さらに，現在の基盤においても仮想機械の配置をポリシで記述することが可能であるが，より安全で高度な資源管理方式についても研究していく．最後に，現在の仮想機械は高度な障害検出機構や高速なスナップショット機構など，信頼性や可用性という観点からはまだまだ十分なインタフェースを提供しているとはいえない．仮想機械の改良についても進めていく．

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．また，実験に協力していただいた豊橋技術科学大学の廣津登志夫准教授および関係各位，産業技術総合研究所の須崎有康氏，八木豊志樹氏に感謝する．

参 考 文 献

1) Keeton, K., Beyer, D., Brau, E., Merchant, A., Santos, C. and Zhang, A.: On the Road to Recovery: Restoring Data after Disasters, *ACM EuroSys'06*, pp.235–248 (2006).

2) Lorch, J.R., Adya, A., Bolosky, W.J., Chaiken, R., Douceur, J.R. and Howell, J.: The SMART Way to Migrate Replicated Stateful Services, *ACM EuroSys'06*, pp.103–115 (2006).
3) 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, *ACM SOSP'03*, pp.164–177 (2003).
4) Potter, R.: One-Click Distribution of Preconfigured Linux Runtime State, *USENIX VM'04 WiPs* (2004).
5) Sugiki, A., Yamatozaki, K., Potter, R. and Kato, K.: A Platform for Cooperative Server Backups based on Virtual Machines, *Int'l Service Availability Symp.'08*, pp.129–141 (2008).
6) 小磯知之，阿部洋丈，池嶋　俊，石川宗寿：サステーナブルサービスのための基盤ツールキットの設計，情報処理学会論文誌：コンピューティングシステム，Vol.48, No.SIG3 (ACS 17), pp.13–26 (2007).
7) VMware Inc.: VMware High Availability. http://www.vmware.com/products/vi/vc/ha.html
8) Bressoud, T.C. and Schneider, F.B.: Hypervisor-Based Fault Tolerance, *ACM TOCS*, Vol.14, No.1, pp.80–107 (1996).
9) Defago, X., Schiper, A. and Sergent, N.: Semi-Passive Replication, *IEEE SRDS'98*, pp.43–50 (1998).
10) Reiser, H.P. and Kapitza, R.: Hypervisor-Based Efficient Proactive Recovery, *IEEE SRDS'07*, pp.83–92 (2007).
11) Stodden, D.: Semi-active Workload Replication and Live Migration with Paravirtual Machines, *Xen Summit, Spring 2007* (2007).
12) Cully, B., Lefebvre, G., Meyer, D., Feeley, M., Hutchinson, N. and Warfield, A.: Remus: High Availability via Asynchronous Virtual Machine Replication, *USENIX NSDI'08*, pp.161–174 (2008).
13) van Renesse, R., Minsky, Y. and Hayden, M.: A Gossip-style Failure Detection Service, *IFIP Middleware'98*, pp.55–70 (1998).
14) Stoica, I., Morris, R., Karger, D., Kaashoek, M.F. and Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications, *ACM SIGCOMM'01*, pp.149–160 (2001).
15) Lamport, L.: The Part-time Parliament, *ACM TOCS*, Vol.16, No.2, pp.133–169 (1998).
16) Lamport, L.: Paxos Made Simple, *ACM SIGACT News*, Vol.32, No.4, pp.51–58 (2001).
17) Mazières, D.: Paxos Made Practical (2007). http://www.scs.stanford.edu/07wi-cs244b/sched/readings/paxos.pdf
18) Castro, M. and Liskov, B.: Practical Byzantine Fault Tolerance, *USENIX OSDI'99*, pp.173–186 (1999).
19) Lamport, L. and Massa, M.: Cheap Paxos, *IEEE/IFIP DSN'04*, pp.307–314 (2004).
20) Perkins, C.: RFC 2002 — IP Mobility Support (1996).
21) Vixie, P., Thomson, S., Rekhter, Y. and Bound, J.: RFC 2136 — Dynamic Updates in the Domain Name System (1997).
22) Onoue, K., Oyama, Y. and Yonezawa, A.: A Virtual Machine Migration System Based on a CPU Emulator, *IEEE VTDC'06*, pp.3–3 (2006).
23) Bradford, R., Kotsovinos, E., Feldmann, A. and Schiöberg, H.: Live Wide-Area Migration of Virtual Machines Including Local Persistent State, *ACM/USENIX VEE'07*, pp.169–179 (2007).
24) Welsh, M., Culler, D. and Brewer, E.: SEDA: An Architecture for Well-Conditioned, Scalable Internet Services, *ACM SOSP'01*, pp.230–243 (2001).
25) Krasnyansky, M. and Yevmenkin, M.: Universal TUN/TAP Driver. http://vtun.sourceforge.net/tun/
26) Tridgell, A.: dbench. http://samba.org/ftp/triage/dbench/
27) Sapuntzakis, C.P., Chandra, R., Pfaff, B., Chow, J., Lam, M.S. and Rosenblum, M.: Optimizing the Migration of Virtual Computers, *USENIX OSDI'02*, pp.377–390 (2002).
28) Cully, B. and Warfield, A.: SecondSite: Disaster Protection for the Common Server, *USENIX HotDep'06* (2006).
29) Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I. and Warfield, A.: Live Migration of Virtual Machines, *ACM/USENIX NSDI 2005*, pp.273–286 (2005).
30) 田村芳明，佐藤孝治，木原誠司，盛合　敏：仮想マシン間の同期による高可用クラスタリング方式の提案，情報処理学会研究会報告（2007–OS–105），pp.71–78 (2007).
31) Sundararaj, A.I., Gupta, A. and Dinda, P.A.: Increasing Application Performance in Virtual Environments through Run-time Inference and Adaptation, *IEEE HPDC'05*, pp.47–58 (2005).
32) Awadallah, A. and Rosenblum, M.: The vMatrix: Server Switching, *IEEE FTDCS'04*, pp.110–118 (2004).
33) Amazon: Amazon Elastic Compute Cloud (2007). http://aws.amazon.com/
34) Wolman, A., Voelker, M., Sharma, N., Cardwell, N., Karlin, A. and Levy, H.M.: On the Scale and Performance of Cooperative Web Proxy Caching, *ACM SOSP'99*, pp.16–31 (1999).
35) Haeberlen, A., Mislove, A. and Druschel, P.: Glacier: Highly Durable, Decentralized Storage Despite Massive Correlated Failures, *USENIX NSDI'05*, pp.143–158

(2005).

(平成 20 年 7 月 23 日受付)
(平成 20 年 11 月 15 日採録)

杉木　章義（正会員）

2002 年電気通信大学情報工学科卒業．2007 年電気通信大学大学院電気通信学研究科情報工学専攻博士後期課程修了．博士（工学）．2007 年 4 月より科学技術振興機構 CREST 研究員，現在に至る．仮想計算機を利用した分散システム，ウェブサーバの性能改善に関する研究に従事．ACM，IEEE-CS 各会員．

大和﨑　啓

2008 年 3 月筑波大学第三学群情報学類卒業．同年 4 月より筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻博士前期課程に入学，現在に至る．分散システムに興味を持つ．

加藤　和彦（正会員）

1962 年生まれ．1985 年筑波大学第三学群情報学類卒業．1989 年東京大学大学院理学系研究科情報科学専攻中退．1992 年博士（理学）（東京大学大学院理学系研究科）．1989 年東京大学理学部情報科学科助手，1993 年筑波大学電子・情報工学系講師，1996 年同助教授，2004 年筑波大学大学院システム情報工学研究科教授，現在に至る．オペレーティングシステム，分散システム，仮想計算環境，セキュリティに興味を持つ．1990 年情報処理学会学術奨励賞，1992 年同研究賞，2005 年同論文賞，2004 年日本ソフトウェア科学会論文賞，各受賞．