Vol.2020-DPS-182 No.57
Vol.2020-CSEC-88 No.57
2020/3/13

データの安全性を確保する災害対応型分散バックアップシステム

飯島 貴政[1,a)]　串田 高幸[1]

概要：自然災害が発生した場合のデータセンターへの影響は地震や台風の直接的な損害に加え, 電力供給会社による計画停電, 土砂崩れによる電線の切断の間接的な損害が考えられる．この手法では 1 分ごとに災害の種類名を SNS 上で検索し，最新の情報のモデル，過去に災害が発生した時刻の前後で前述したデータのモデルをピアソンの相関関数によって類似率を計算し自動で構成とデータを複製することで耐障害性を実現する分散バックアップ手法を提案する．既存のバックアップ手法と比較してバックアップが完了するまでの時間の削減を目的とした．既存のバックアップ手法と提案手法での災害発生からバックアップが完了するまでの時間面と費用面での比較を行うことで評価を行った. 結果, 既存の手法に比べ災害発生からバックアップ開始から終了までの作業時間が約 51.8%削減された．また，期間を設定し評価を行った際に約 49.9%コストが削減できた．

1. はじめに

自然災害が発生した場合のデータセンターへの影響は地震, 台風の直接的な損害に加え, 電力供給会社による計画停電, 土砂災害による生活インフラストラクチャの間接的な損害が考えられる. 総務省の調査によると 2020 年現在電子マネーのような電子決済手段や医療, 行政, 電力のシステムのように官民企業問わず日常のインフラストラクチャには ICT が活用されている [1]．これはシステムが動作しているデータセンターが自然災害にあった際に生活基盤が不安定になることを意味する．本論文で取り扱うクラウドコンピューティングの利点として, データを地理的に分散させて耐障害性をもたせることが挙げられるが, 実際にはクラウドのアーキテクチャを設計する際に考慮すべき問題である地理的な問題に適切に対応できる耐障害性の設計をしないことが散見される. この問題は Amazon Web Services や Google Cloud Platform といったベンダー固有のパブリッククラウドの各機能の使用方法を熟知する必要があり, 高可用なクラウドシステムを導入できる人材不足や耐障害性を含む高可用性のクラウドシステムの学習及び理解に難度が高いことが原因としてあげられる．

本論文では分散バックアップ手法, Disaster Response Backup and Recovery System(DRBRS) を提案する．本論文の手法は毎分 Twitter から災害に関連するツイート数を入手する． 過去のデータに基づくしきい値を超過時に別リージョンへの構成及びデータの複製を開始する．人間が介在する事なく自動的にデータが消失を防ぐために構成及びデータを複製する分散バックアップ手法である．本論文においてクラウドリソースとはクラウド上に存在する構成情報及びデータと定義する．またクラウドの構成情報とはクラウドリソースのうち，データを含まないクラウドサービスの枠組みのみを取り扱うものと定義する．2 章では本研究に関連するバックアップ手法及びデータの以降に関連する研究を列挙する．3 章では本研究における災害検出モデル及びクラウドリソースの複製のプロセスを提案する．4 章ではこれらを実際のクラウドリソースに対し実装について記述する．5 章では前述した既存のバックアップ手法との速度面と費用面での比較を行うことで評価する．

[1] 東京工科大学コンピュータサイエンス学部
CDSL, TUT, Hachiouji, Tokyo 101–0062, Japan
[a)] c0116023c2@edu.teu.ac.jp

2. 関連研究

Disaster Recovery for System Architecture Using Cloud Computing ではパブリッククラウドでの一般的な障害対策について述べられている [2]．地震や津波, 台風の自然災害が発生した際のクラウドリソースについて既存のバックアップ手法には, 以下の 2 つがある．

(1) 災害が発生してから人間が手動でリソースを別リージョンに複製する．
(2) 事前に別リージョンにリソースをプロビジョニングする

手法 (1) は手動であるためにバックアップ元のリソースの

規模が大きくなるほど多くの時間を要する. また, データセンターのオペレーターが席を離れている間に震災が発生した場合メンテナンスの着手までに時間がかかる. 本手法ではバックアップの開始を自動化することで，人間が駆けつけられない事態においてもシステムを保護できる．手法 (2) は恒常時よりバックアップを維持させるために余剰なコストが発生する.

- CloudNet クラウドリソースのバックアップ手法において, CloudNet は VPC を用いて同じ VPN 内において稼働中のシステムをそのままマイグレーションできる手法である [3]．この手法は WAN 上において VPLS を用いてリソースの Cloud-to-Cloud マイグレーションを行っている．論文中で扱われているバックアップをするリソースが少ないこと及び VPC ネットワークの構成そのものをバックアップをしていないことが挙げられるため，クラウドの規模が大きくなった際に適していない.
- Cloud Security Issues この論文では, 既存のクラウドコンピューティングにおいて自然災害のような予期し得ない通信の遮断がクラウド全体での課題をしている [4]. 今回のアプローチでは災害は予期し得ないが可能な限り素早くバックアップを取る必要があるため，人間のオペレーションを待たないという点において既存のクラウドコンピューティングにおける課題を一部解決した.
- Disaster Recovery as a Cloud Service. Timothy らは DR をクラウドで実行した場合, システムが 99%稼働していると仮定した際にオンプレミス環境と比較して VM の場合は 0.15 倍, データウェアハウスの場合は 0.88 倍のコストで維持できる事が示されている [5].

これらのことから本研究では既存のクラウドバックアップの手法と比較し，バックアップ維持コストが低いクラウドからクラウドへの DR 手法を考案した．本研究はメジャーなパブリッククラウドをターゲットとして災害が起きた際に自動的に実行することで RTO(Recovery Time Objective, 目標復旧時間) の短縮をした.

3. 設計

本手法では過去の災害発生時の地震, 台風の災害種別名を含むツイート数を保存する災害検出モデルを生成し，1 分毎に災害種別名を含むツイート数を取得するモデルと比較し，ピアソンの定理で類似率を計算する．類似率がしきい値を超過した場合にクラウドリソースの複製を行う．リソースの複製は構成管理ツールを用いてクラウドの構成を取得し，一時的に DRBRS のサーバに保存する．すべての構成が取り終えた段階で別のリージョンに全く同じ構成を配置する．その後データベースや VM(Virtual Machine) のデータがあるものについてはクラウドリソース間でデータマイグレーションを行う．気象庁の気象庁震度階級関連解説表によると停電が発生する可能性について，「震度 5 弱程度以上の揺れがあった地域では、断水、停電が発生することがある。」としている [6]．また，電話等通信については以下のように定義している」[6].

- 「地震災害の発生時、揺れの強い地域やその周辺の地域において、電話・インターネット等による安否確認、見舞い、問合せが増加し、電話等がつながりにくい状況（ふくそう）が起こることがある」
- そのための対策として、震度 6 弱程度以上の揺れがあった地震の災害の発生時に、通信事業者により災害用伝言ダイヤルや災害用伝言板の提供が行われる.

これらの情報を元に本論文ではデータセンターに影響を及ぼす地震=震災は最大震度 5 弱以上のものと定義する.

3.1 災害検出に用いるデータの情報源

災害の前後の Twitter のデータとして Twitter 上での災害名キーワードを検索する. 例として”地震”を検索した結果から過去に災害発生した時間周辺のツイートを災害発生の前 X 時間, 後 Y 時間のツイートを取得する．条件として Twitter 上でのリツイートやお気に入りなどを Twitter の検索クエリである vertical news を利用してエンゲージメントが多いものを収集した後ソートし, データセットとした. 1 ツイートに含まれる要素を表 1 に示す. ツイートの情報上では geo から地理情報が求められる．しかし実際にはツイートに位置情報を付加しているツイートはサンプルとして災害の日に取ってきたトップツイート 15 万件のうち, わずか 1214 件であり，全体の割合としては 0.9%に満たない数量であった．そのため，今回は位置情報はデータとして使用せず，ツイート日時とツイート数の関係をデータとして使用した.

表 1 1 ツイートに含まれる要素

要素名	説明
text	ツイートの本文
date	ツイートの日時
retweets	リツイートされた数
favorites	お気に入りに追加された数
hashtags	”#”から始まるタグ
geo	地理情報が付与されていた場合の緯度経度

3.2 災害検出モデル

本研究における災害検出に用いるデータは現在の災害ツイートをリアルタイムで取得する Current Steram と過去の災害における Current Stream を保存する Disaster Occur Model である.

3.2.1 災害情報取得ストリーム:Current Stream

災害のデータの範囲においては地震の場合は地震直後から，台風の場合は 1 週間から 3 日前，ツイート数が増加す

るタイミングが災害によって異なる．そのため，モデルの定義においては災害発生の前後のデータの範囲をそれぞれX,Y時間とし，今回の実験，評価においては地震をターゲットとした．地震は災害が発生するまで災害を予期できないため，災害が発生してからツイート数が顕著に増加する．今回の実験, 評価ではデータ範囲である X,Y をそれぞれ 1 時間と 2 時間と設定した．現状の災害状態を収集するため, 1 分毎に”地震”を検索した結果を取得し, データとして保存する. これを CurrentStream と呼ぶ．CurrentStream の取得時から最新 X+Y 時間分の時間の経過とツイート数の関係を図 1 に示す．

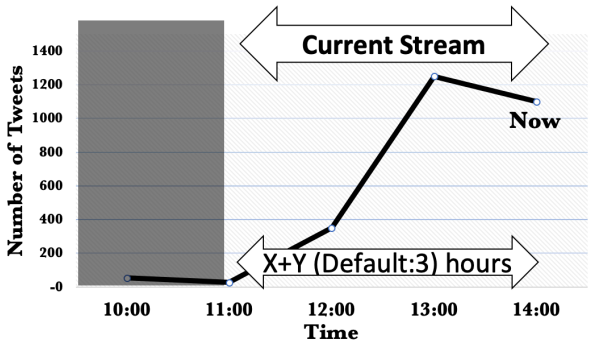

図 1　Current Stream の範囲

3.2.2　災害モデル:Disaster Occur Model

災害発生時の CurrentStream を災害モデルとして用いる．Disaster Occur Model は災害時における Current Stream を災害発生時の X 時間前から Y 時間後までを保存したものである．ただし，災害時のデータとして災害の前後二週間のデータを Disaster Occur Model とは別に保存する．

- H は災害発生の時刻のうち時間要素, M は災害発生の時刻のうち分要素である.
- データの収集の条件により N-13 日 H 時 M 分より N+13 日 H 時 M 分までのツイートが保存されている.
- N 日 H-X 時 M 分から N 日 H+Y 時 M 分の 3 時間の災害発生時の日付の経過とツイート数の関係をモデルとする. これを Disaster Occur Model(DOM) と呼ぶ.

図 2 は本研究において評価に使用した値である X=1,Y=2 とした時の DOM の範囲の例である．グレーにマスキングされている部分はデータとしては収集しているが DOM の要素としては扱わない.

3.3　Disaster Occur Model の生成例

以下は Disaster Occur Model 適用した例である．図 3 では災害発生前後のツイート数の推移において災害発生時におけるツイート数の急激な増加がある．これは地震の揺れがあった際に地震速報ツイートやその再投稿である．その後も引き続き多数ツイートされている中で，6 時間後にもう一度スパイクしている時間がある．これは政府機関が震災に対して会見を開催したのち二次災害について頻繁に情

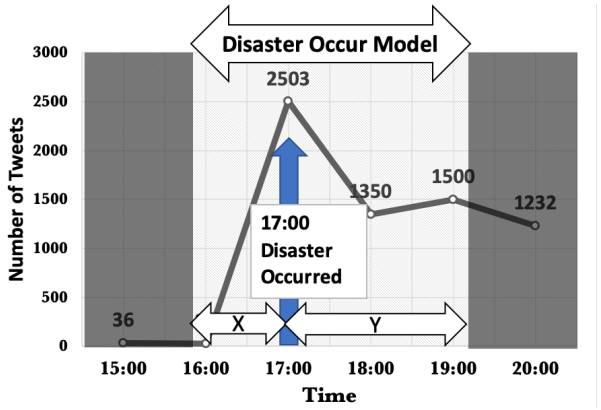

図 2　Disaster Occur Model の範囲

報交換している様子である．小さな地震では 1 のスパイクのみ，最大震度 6 以上の地震では①及び②の急激なツイートの増加が確認されることが多かった．DRBRS では主に①のスパイクを震災の判定に用いた．これは 1 のスパイクが実際の震災発生であり, ②のタイミングでは①のタイミングによるバックアップ生成ができているためである．

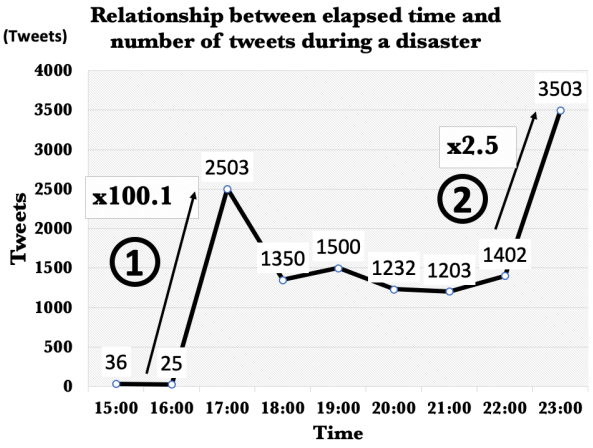

図 3　震災発生からのツイート数増加の例

時刻 t における 1 時間前からのツイート数の増加度 r(倍) を以下の数式で求めた.n(t) は時刻 t におけるツイート数である.

$$ratio = n(t)/n(t-1)$$

また, その計算結果を図 3 の時間軸と合致させたものを図 4 に示す．その結果図 2 と比べ①のスパイクでは 1000 倍になったが②のスパイクでは 2.5 倍となり，①のスパイクのみが検出可能になった．これにより同じ災害のツイートの盛り上がりを複数の災害として検知する誤動作を防ぐことができる．

CurrentStream が Disaster Occur Model に 80%以上類似することが災害が発生したと定義する. 類似度の計算にはピアソンの相関関数では線形な 2 つのグラフの類似度を算出できる [7]. この手法では 2 つのグラフが線形だった場合は有効だが, 災害の後のツイート数を見た際に増加だけではなく減少部分もあるため 2 次,3 次曲線として比較をする手

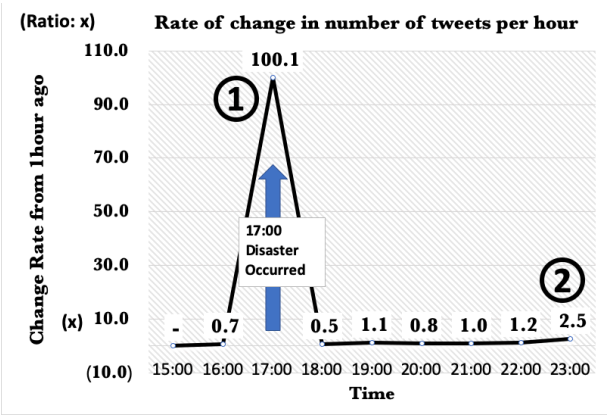

図 4　timeline: training

法であるスピアマンの手法と MIC(Maximun Information Coefficient) の 3 手法を比較したものを表 2 に示す [8][9]. 比較での指標として，1 つの Disaster Occur Model が存在する場合に,1 分ごとに 3 時間の Current Stream との類似度を計算に必要な回数である 180 計算での処理時間を用いた.

表 2　1 ツイートに含まれる要素

手法名	線形相関	2 次関数	3 次関数	180 計算あたりの計算秒数
ピアソン	○	X	X	0.9(s)
スピアマン	○	○	△	2.7(s)
MIC	○	○	○	30.3(s)

計算コストを考慮しなければ MIC がすべてのグラフについて正しく相関を取れているが，ピアソンの手法と比べた場合約 20 倍もの計算コストである. また,DOM と CS を用いて 365 日分の CS で後述の精度テストをしたところ，ピアソンの手法の方が 12%ほど精度が高い結果になった．これは災害の後のツイートの減少までの相関を正確に取りすぎてしまった結果，DOM と完全一致ないしかなり近い CS でないと検出できなかったことが挙げられる．Discriminator の評価をする．災害モデルに登録されていない自然災害時の Twitter を擬似的に Current Stream Model に流し込み, 実際の災害発生時刻と DRBRS が検知した災害発生時刻を測定する．また DRBRS が災害を検知してから別リージョンへのリソースのバックアップを完了するまでの時間を計測する. 今回 Disaster Occur Model に保存したモデルは表 3 に示すの 2 つの地震である. 擬似的に Twitter のツイート

表 3　評価に用いた Disaster Occur Model の日付

地震名	発生日時
東日本大震災	2011-03-11 14:46:18
熊本地震	2016-04-14 21:26:23

を再現するため, ランダムな日時の CurrentStreamModel と災害に近い日付のダミー Current Stream をそれぞれ 5 つ計 10 個生成した. これらを Discriminator に送信し, 災害が起きたかの判定および, 災害だと判定した場合は実際の発生時刻との差分を調査する.

3.3.1　実験に用いたダミーデータ

ソースコード 1　dummylist.json

```
{
  "東日本大震災-1時間": "2011-03-11 13:46:18",
  "東日本大震災から 1 年後": "2012-03-11 14:46:18",
  "熊本地震-1時間": "2016-04-14 20:26:23",
  "大阪府北部地震": "2018-06-18 07:58:34",
  "北海道胆振東部地震": "2018-09-06 03:07:59",
  "山形県沖地震": "2019-06-18 22:22:19",
  "dummy_test1": "2020-01-13 12:15:23",
  "dummy_test2": "2020-01-13 23:15:23",
  "dummy_test3": "2018-05-25 12:15:23",
  "dummy_test4": "2016-01-13 15:15:23",
  "dummy_test5": "2019-11-06 21:15:23"
}
```

これらを用いて Discriminator にツイートを注入した結果が表 4 である．表 4 では地震名, 相関関数の計算による Disaster Occur Model との類似度，80%をしきい値とした際の災害かそうでないかの Discriminator の判定結果, 最後に実際の世界で地震が発生していたかを人間が日本気象協会の過去の地震情報にて確認し，Discriminator の判定結果があっていれば○， 間違っていれば X とした.

表 4　ピアソン相関での評価結果

地震名	DOM との類似度	災害 orNot	判定
東日本大震災-1 時間	-4.62%	Not 災害	○
東日本大震災から 1 年後	49.96%	Not 災害	○
熊本地震-1 時間	11.55%	Not 災害	○
大阪府北部地震	93.50%	災害	○
北海道胆振東部地震	97.27%	災害	○
山形県沖地震	97.74%	災害	○
dummy_test1	-11.67%	Not 災害	○
dummy_test2	-5.41%	Not 災害	○
dummy_test3	17.36%	Not 災害	○
dummy_test4	-4.46%	Not 災害	○
dummy_test5	-3.58%	Not 災害	○

本評価では精度が 100%であった．表 4 では災害から 1 年後の東日本大震災の相似率が 49.96%ということから，同日に過去に災害があった日においてもご認識することなく災害を判定できた．同じように，365 日でテストをすると精度は 87%であった．また，地震発生時における DOM との類似度の平均が 89.6%であった. これらの結果から Discrimitor のしきい値を 85%と設定した．図 5 に twitter から discriminator にデータが入力され，複製プロセスがトリガーされるまでのシステムフローを示す.

(1) 2 章で述べた災害検出モデルの情報源となる Twitter から毎分災害名でクエリしたツイートをサーバに保存する.

(2) そして直近 3 時間のデータ (Current Stream) と過去の災害から学習した災害モデル (Disaster Occur Model) を Discrimitor で災害かどうかを判別する.

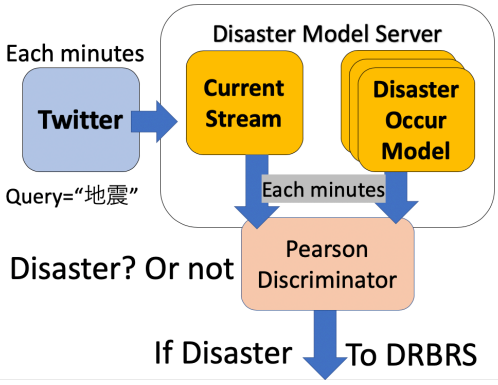

図 5　災害検知フロー

3.4　災害が発生したと判定された場合の動作

3 章の災害検出モデルを用いて, 災害を検出した場合にはコンピューターリソースの複製を開始する. 複製はクラウド上の構成情報およびそれぞれの構成要素の中のデータそのものに対して行う. 本研究で実験及び評価に使用する環境は Google Cloud Platform 上での Virtual Machine(VM) である Google Compute Engine(GCE), コンテナオーケストラレーションのアーキテクチャである Google Kubernetes Engine(GKE) および Key Value Store 型のデータストレージである Google Storage である. 以下に Discriminator によって DRBRS がトリガーされた後のフローである.

(1) 災害が発生している結果になった場合にターゲットのクラウドの構成を Configration Store Server に保存する.
(2) 保存された構成をもとに別リージョンにリソースを作成をリクエストする.
(3) リクエストした結果別リージョンに空のリソースが作成される.
(4) オリジナルのリソースから新しく作成したリージョンにデータの内容をマイグレーションする.

図 6 に上記のフロー図を示す

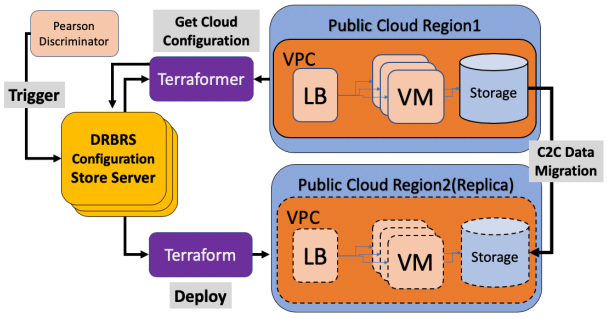

図 6　Disaster Response Backup and Recovery System

4.　実装

DRBRS は災害判定の元データを保存する Disaster Model Store Server,Disaster Occur Model と Current Stream を入力とし, 災害かそうでないかを判別する Discriminator, バックアップの対象となるクラウドのアーキテクチャを保存する Configuration Store Server から構成される.

4.1　Disaster Model Store Server

Disaster Model Store Server は過去の震災の災害検出モデル及び Twitter から 1 分毎に収集したデータを保存する. データは変更されることはなく, 高速な読み書きのみが要求されるためデータベースは NoSQL である MongoDB を採用した. また Disaster Model Store Server 自体に耐障害性を持たせるために 3 つの地理的に分散したリージョンに 2 つ計 6 つのデータベースを用いる. 各リージョン間は 1 時間毎にデータを同期させる. データの収集は Twitter からのデータが入手しやすく MongoDB へ高速に書き込みができる Python3.6 を採用した.

4.2　Architecture Store Server

Architecture Store Server はバックアップをする対象のクラウドのアーキテクチャをテキスト形式の構成ファイルとして保存する一時的なストレージである. バックアップ元のデータ本体は保存しない. デフォルトでは 2 週間の間バックアップ元の構成ファイルを保存する. これにより, クラウドの管理者は必要に応じて 2 週間以内であれば即座にバックアップを更に冗長化することができる. データベースは MongoDB を採用した. VM 間のデータトランスファーは Google のライブマイグレーションを用いた.

また既存の技術としてパブリッククラウドから現状の構成をエクスポートする terraformer, 前述のエクスポートしたファイルからパブリッククラウドのリソースを作成する Terraform を使用している [10][11].

5.　評価

DRBRS のクラウドレプリケーションにかかる時間を検証する. また時間から得られるコストの削減を計算する.

5.1　評価環境

Disaster Model Store Server

研究室内の 2 つの ESXi サーバーを別のリージョンと見立ててそれぞれに CoreOS をインストールした VM を用意した.VM 上で Docker で MongoDB を立ち上げ, 永続化ストレージはそれぞれのサーバーの NAS のマウントポイントとした.

Configuration Store Server

研究室内の 2 つのサーバーを別のリージョンと見立て, それぞれに CoreOS をインストールした. Docker で MongoDB を立ち上げ, 永続化ストレージはそれぞれの NAS のマウントポイントとした.

バックアップ対象のパブリッククラウド

Google Cloud Platform 上に Google Compute Engine x2, Google DNS,Google Storage の 3 つのリソースを配置した．Python 上で動作する動的なグラフに有用な WEB サーバライブラリである Dash を用いて pymongo 上に保存されたモデルの表示した．

5.2 評価結果

Discriminator のトリガーをエミュレートして DRBRS の挙動および，それぞれのリソースのレプリケーションに掛かる時間を計測した．また既存の手法として Google gcloudCLI を用いてリソースを複製した際の時間と比較する．また，地震と地震の間の時間に掛かる VM コストをシュミレーションすることで既存の DR 手法とのコスト面の比較を行う．

クラウドリソースを人間が複製する既存の手法と，DRBRS を複製に要する時間で比較した．

表 5 クラウドリソースの複製にかかる時間

リソース名	既存の手法 (分:秒)	DRBRS(分:秒)
GCE	67:14	21:15
DNS	24:11	01:10
Storage	70:15	60:12
合計	:171:40	82:37

作業時間の差は既存の手法では人間が gcloudCLI と WEB 上のコンソールを用いて処理を実行しているのに対し DRBRS ではプログラムによって自動的に処理されるため高速化されている．本評価においては既存の手法と比較し 51.8%を削減できたといえる．

コスト面については GCP での GCE のランニングコストはリージョンによって異なる．us-central1 での 1 時間あたりのコストは$0.08 であり，これは 1 ヶ月では$58.32/month である．asia-northeast1 では 1 時間あたり$0.103,1 ヶ月では$74.84/month である．

震災と震災の間が 1 2 ヶ月に 1 回と仮定し，メインのリージョンを us-central とした時にレプリカを asia-northeast1 に配置したとする．その際に DRBRS の複製が表 5 の時間で終わった後，元のインスタンスをシャットダウンしたと仮定する．また，コストの比較においては時間単位での料金となっている GCE のみでの計算とする．コストの計算は一時間あたりのコスト*バックアップにかかる時間=バックアップに要するコストとした．

表 6 既存のレプリカ手法と DRBRS のバックアップ時のコストの比較

リソース名	一回のバックアップを実行中にかかるコスト
手動でのレプリケーション	($0.08+$0.103)*(67.23/60) =$0.205
DRBRS	($0.08+$0.103)*(21.25/60) = $0.064

バックアップ中の時間のみの差は$0.141 である．2 ヶ月間テストをし，1 ヶ月目の末に災害が発生したとする．もともと 2 台のレプリカを維持した場合には ($58.32+$74.84)*2 ヶ月=$266.32 である.DRBRS を用いた場合には 1 ヶ月目の VM コスト$58.32+災害が起きたあとの VM コスト$74.84+マイグレーション中の両方のコスト$0.141=$133.30 である．VM1 台につき DRBRS を使うことで既存にレプリカを設置する手法と比べ, コストを 49.94%節約したといえる．

6. 議論

既存のクラウド上にある全てのリソースが耐障害性を考えられていない場合にはこの提案手法は有効である. しかしシステムの一部でもすでに別リージョンでのプロビジョニングが組み込まれている場合に DRBRS は余分な冗長化をするためコストが発生する. これはバックアップを常に 1 つ以上用意するという点では効果的である．また, 災害を検出するプログラムの精度においては単純なピアソンの相関関数を使った場合，1 年間の災害ツイート検索結果, 計 365 サンプルでの検証では 87%の精度で正しい判定を出力できた．また，Twitter で取得した情報の信頼度を確実にするため，情報源を増やしてメディアをクロスチェックする工夫が必要である．結果, 既存のバックアップ手法に比べ災害発生をシステムから検知できるようになったため，災害発生からのリカバリーを高速化することができた．今回は実験環境に GCP を用いたが，今後は OpenStack や AWS,Azure で実験, 評価する必要がある．本手法は地震の後に余震や 2 次災害が発生する前にバックアップを他のリージョンに移す手法であるため，1 回目の地震で電線，ネット回線が使えない状況になった際，バックアップが作成できない．これに対し, 本手法で用いた Disaster Occur Model を機械学習に入れて災害の予測が可能かどうかを検証する必要がある．実際のサービスにおいてはクライアントが絶えず対象のクラウドにアクセスすることが想定されるため，どのタイミングで複製し，切り替えるべきであるかを調査する必要がある．また, クラウドリソースを複製した後に元のサーバーに対してアクセスしてきたクライアントに対してのリダイレクトを自動設定する必要がある．

7. 終わりに

この研究では 2 章にて挙げた現状のクラウドのバックアップについて課題であった災害時に人間がオペレーションする内容を自動化し，クラウドリソースのバックアップにかかる時間を 49.9%短縮できた．また，既存のクラウド上で災害が起きる前よりリソースを複製する手法と比較して 51.8%削減できた．3 章で実装した Discriminator については地震が発生したことを 87%の精度で取得できている．DRBRS のトリガーとして用いるだけではなく，他の用途に使うこともできる．例えば災害情報を提供する WEB サーバーにおいて，災害後の大量のアクセスに備えて予め追加の Docker コンテナを起動させるトリガーになる．その結果サービスの可用性が上がるといえる．現在の

クラウドリソースのバックアップの手法は安全性を重視する場合, コストが増加していた．本研究ではコストが通常時とほぼ変化せず災害時にバックアップを作成する手法を提案した．これからのクラウドの冗長化のもうひとつの選択肢として Disaster Response Backup and Recovery System(DRBRS) を提案した．

参考文献

[1] 総務省: ICT によるイノベーションと新たなエコノミー形成に関する調査研究, https://www.soumu.go.jp/johotsusintokei/linkdata/h30_02_houkoku.pdf.
[2] Pokharel, M., Lee, S. and Park, J. S.: Disaster Recovery for System Architecture Using Cloud Computing, *2010 10th IEEE/IPSJ International Symposium on Applications and the Internet*, pp. 304–307 (online), DOI: 10.1109/SAINT.2010.23 (2010).
[3] Wood, T., Shenoy, P. J., Gerber, A., van der Merwe, J. E. and Ramakrishnan, K. K.: The Case for Enterprise-Ready Virtual Private Clouds., *HotCloud* (2009).
[4] Kandukuri, B. R., V., R. P. and Rakshit, A.: Cloud Security Issues, *2009 IEEE International Conference on Services Computing*, pp. 517–520 (online), DOI: 10.1109/SCC.2009.84 (2009).
[5] Wood, T., Cecchet, E., Ramakrishnan, K. K., Shenoy, P. J., van der Merwe, J. E. and Venkataramani, A.: Disaster recovery as a cloud service: economic benefits & deployment challenges., *HotCloud*, Vol. 10, pp. 8–15 (2010).
[6] 気象庁: 気象庁震度階級関連解説表, https://www.jma.go.jp/jma/kishou/know/shindo/kaisetsu.html.
[7] Benesty, J., Chen, J., Huang, Y. and Cohen, I.: Pearson correlation coefficient, *Noise reduction in speech processing*, Springer, pp. 1–4 (2009).
[8] : *Spearman Rank Correlation Coefficient*, pp. 502–505, Springer New York (2008).
[9] Speed, T.: A Correlation for the 21st Century, *Science*, Vol. 334, No. 6062, pp. 1502–1503 (online), DOI: 10.1126/science.1215894 (2011).
[10] terraform: Terraform, https://www.terraform.io/.
[11] Platform, G. C.: terraformer, https://github.com/GoogleCloudPlatform/terraformer.