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などの直接的な損害に加え, 電力供給会社による計画停電など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人間の災害時のオペレーションを待たずに災害時に自動で同じ構成の同じ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当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, 既存のバックアップ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 53%削減された．期間を設定し評価を行った際に 49%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や行政, 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]. 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．本論文の手法は災害がおき始めたときに SNS のトレンドから情報を入手し, 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リソース複製を開始し, 人間が介在する事なく自動的にデータを維持するよう複製する分散バックアップ手法である．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を行う.

[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)] C0116023

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

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, 深夜, 休日でデータセンターを管理する側の人間が休息を取っている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. 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．比較的に移動するリソースが少ないこと及びネットワークの構成はバックアップをしていないが挙げられるため，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 Cloud Security Issues 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].
- 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].
- Remus Remus では VM の Replication を行う際に Failure model と Pipelined Checkpoints でデータの整合性を確保しつつ高速な転送のモデルを提唱している，本論文においてはクラウドリソースのうち GCE,EC2 の VM インスタンスを複製する際に用いた．

このことから本研究ではパブリッククラウド上からオンプレミス環境にバックアップするのではなくパブリッククラウド上の他リージョンに構成を複製することでクラウド上での機能を損失しないマイグレーション及びオンプレミス環境にバックアップするより維持コスト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．また，地理的に分散したリージョンにクラウドリソースを複製することで恒常的な維持コストを減少させるアーキテクチャ,DRBRS を提案する．

3. 設計

本手法では過去の災害発生時の地震, 津波等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状況を取得し，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) など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．

3.1 災害検出モデル

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, 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 時間のツイートをエンゲージメントが多いもののみを収集し, データセットとした.1 ツイートに含まれる要素を 7 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．そのため今回はツイート日時と数量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

並びに, 現状の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. また, CurrentStream の取得時から最新 3 時間分の時間の経過とツイート数の関係をグラフにしたものを Current Stream Model と呼ぶ. 災害発生時間を N 日 H 時 M 分とする.N は災害発生の日付, 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数である. 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 150000 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 3.1 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.

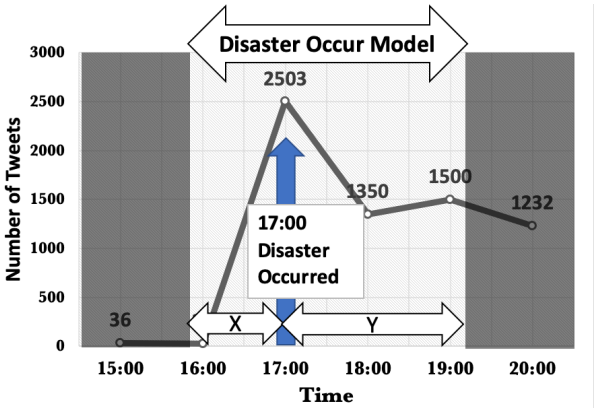

図 1

以下は Disaster Occur Model を適用した例である．図 3.1.1 では災害発生前後のツイート数の推移ではまず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．

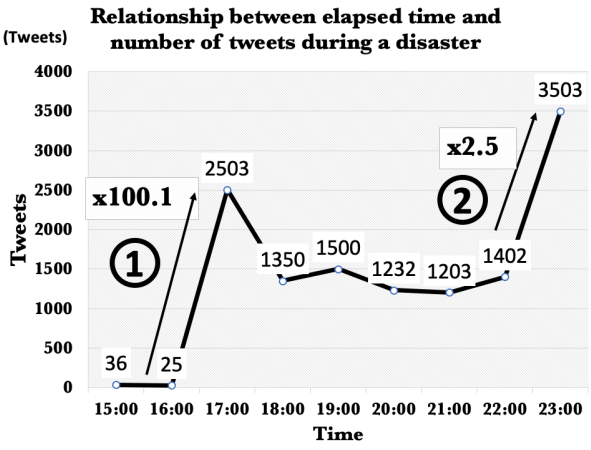

図 2

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.T は時刻 T におけるツイート数である．

$$ratio = n(t)/n(t')$$

また, その計算結果を図 3.1.1 の時間軸と合致させたものを図 3.1 に示す．その結果図**??**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を 2 度別の災害として検知する誤動作を防ぐことができる．

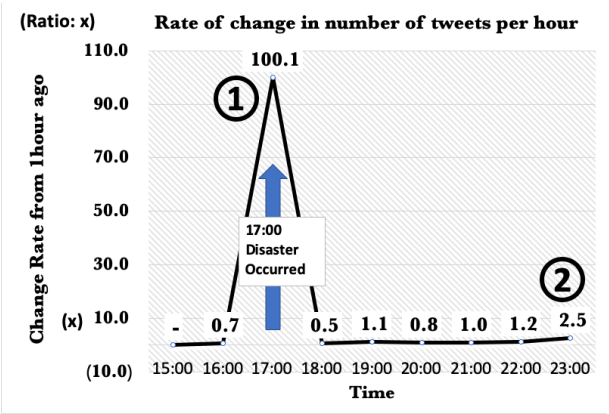

図 3 timeline: training

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 [6]. 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．

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	100 計算あたりの計算秒数
ピアソン	○	X	X	0.5(s)
スピアマン	○	○	△	1.2(s)
MIC	○	○	○	12.2(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が，DOM と CSM を用いて 365 日分の CSM 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は 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 と完全一致ないしはかなり近い CSM 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以下の表の 2 つの地震である．擬似的

表 3 評価に用いた Disaster Occur Model

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

に Twitter のツイートを再現するため, ランダムな日時の CurrentStreamModel と災害に近い日付のダミー Current Stream Model 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. ピアソンの相関関数の計算の他に, 視覚的に震災のツイート数と時間の流れがひと目で理解できるように, WEB サーバーを立ち上げ, Current Stream Model と Disaster Occur Model の折れ線グラフでの類似も参考にした. 横軸が時間の流れ， 縦軸がツイート数である．縦に 2 つグラフがあるうち, 上段が Current Stream Model が 1 分毎に更新される．下段が Disaster Occur Model である．上段は時間の流れとともに右から左へデータが流れるが Disaster Occur Model は変化しない．この WEB ページによりグラフの勾配が重なったときに DRBRS が作動するのが視覚的なインターフェイスによって理解しやすくなる．

図 4 DRBRS Model Monitor

3.1.1 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 3 である．表 3 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判定結果, 最後に実際の世界で地震が発生していたかを確認し，判定結果があっていれば ○， 間違っていれば X とした．

表 4 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．災害から 1 年後の東日本大震災の相似率が 49%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．また，365 日でテストをすると精度は 87%であった．これらの結果から DRBRS はピアソンの手法を用いて以降の複製プロセスをトリガーする．以下に 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を示している．

3.2 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. 図 3 は本研究でのシステムフローを示している.

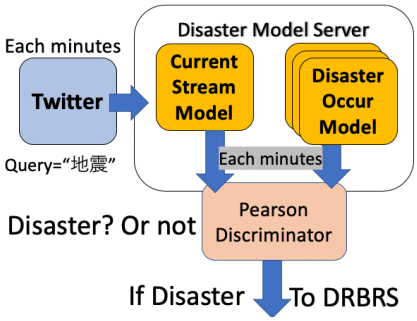

図 5

(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 Model) と過去の災害から学習した災害モデル (Disaster Occur Model) を正規化相関で災害かどうかを判別する.

(3) 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 Architecture Store Server に保存する.

(4) 3 の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.

(5) 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.

(6) 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.

(7) マイグレーションの結果, 全てのリソースが正常にクローンできたことを Validator で確認する.

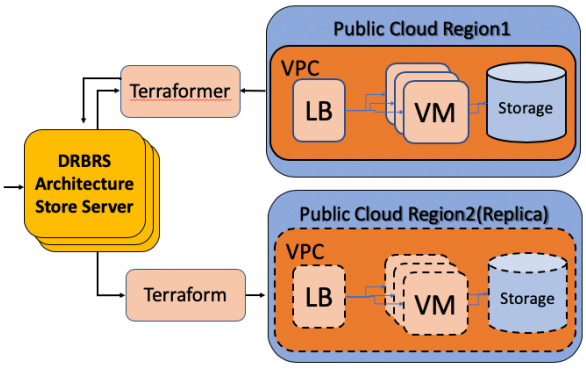

図 6 Disaster Response Backup and Recovery System

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 Model 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 Architecture Store Server, バックアップ元と新規作成したバックアップ先のアーキテクチャ及びリソースのデータ本体部が整合しているかを検証する ValiCloud 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 Architecture Parser の部分で使用している.

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.

Architecture 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.

DRBRS Model Monitor
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．

表 5 クラウドリソースの複製にかかる時間

リソース名	既存の手法での構成複製	DRBRS での構成複製
GCE	1:17:14	00:21:15
DNS	0:24:11	00:01:11
Storage	01:10:15	00:60:15

コスト面については GCP での GCE のランニングコストを以下の表に示す.

表 6 GCE の利用料金

リソース名	一時間あたりの利用料金	一ヶ月あたりの利用料金
us-central1	$0.08	%58.32
asia-northeast1	$0.103	$74.84

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.

表 7 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*171/60 =$0.228
DRBRS	$0.103*171/60 = $0.140

バックアップ中の時間のみの差は$0.78 であり．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$0.121=$133.3 である．VM1 台につき DRBRS を使うことで$133.2 を節約したことになる.

6. おわりに

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当手法は有効である. 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うのみでも 10 サンプル中すべて正しい判定を出力できた．しかし，評価においてダミーのサンプル数が少なかったため，今後はダミーを 100 倍ほど増やして評価する．また，情報源を増やしてメディアをクロスチェックするなどの工夫が必要である. 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 など他のパブリッククラウドで実験しようと思う,

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] Benesty, J., Chen, J., Huang, Y. and Cohen, I.: Pearson correlation coefficient, *Noise reduction in speech processing*, Springer, pp. 1–4 (2009).