(302) ネットワーク

障害につながる状態変化を表現可能な FIT システムの提案

A Proposal of a FIT System Considering State Transitions as a Sign of Failure

土手 貴裕[†1]　前田 香織[†2]　近堂 徹[†3]

Takahiro Dote[†1]　Kaori Maeda[†2]　Tohru Kondo[†3]

[†1] 広島市立大学情報科学部情報工学科　[†2] 広島市立大学大学院情報科学研究科

[†3] 広島大学情報メディア教育研究センター

1 はじめに

ストレージサービスや Web サービスなどの様々なサービスを提供する方法として，クラウド環境を用いることが一般的になっている．そのためクラウドサービスを障害なく安定的に提供することが求められる．クラウド環境における耐障害性に関する研究では，障害発生後に障害を検出し対処するリアクティブな運用管理ではなく，定常状態から外れる状態変化を重大な障害につながる可能性のある異常として検出し，障害発生前に対処するプロアクティブな運用管理を行うアプローチが増えている[1][2]．これらは事前に収集した実稼働環境とは異なるデータを機械学習に用いて異常検出する手法がほとんどであるため，実稼働環境における異常検出の有効性についての検証が必要不可欠である．

一方，実際に稼働するシステムに意図的に障害を注入する FIT(Failure Injection Testing)[3]により耐障害性を向上させるアプローチもある．しかし，現在の FIT ツールでは障害の発生シナリオが限定されており，障害につながるような状態変化を表現することはできない．

そこで，本研究では障害につながる時系列的な状態変化を表現可能な FIT システムを提案する．これにより，障害の誘因となる状態変化を異常として検出する監視システムに対して，実稼働環境における有効性の検証が可能となる．

2 関連研究

2.1 プロアクティブな異常検出方法

クラウド環境を用いた運用システムの振る舞いは複雑なため，異常を検出するための閾値を人が適切に設定するのは難しい．そのため，近年は運用システムの異常検出の手法として，ネットワークログの生成パターンを用いた手法[1]やリソースの使用パターンを用いた手法[2]のように，状態の機械学習に基づいた方法が考えられている．

しかし，これらは実稼働環境で採取したデータを異常検出に使用しているとは限らず，また実稼働環境において異常を正しく検出できるかの動作検証が行われていないため，実稼働環境での異常検出の有効性を示すことが難しい．

2.2 Failure Injection Testing

FIT[3]はクラウド上の運用システムの耐障害性を向上させるために運用システムに意図的に障害を注入し続ける試みである．Netflix は運用システムに対し意図的に障害を発生させた際の運用システムの振る舞いを把握するアプローチを Chaos Engineering[4]と定義している．

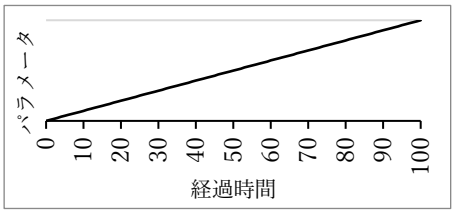
FIT を実現するツール(FIT ツール)にはコンテナレベルで障害を注入する pumba[5]や，インスタンスをランダムに落とす Chaos Monkey[6]などがある．FIT ツールによってクラウド上の運用システムに障害を注入し，そのときの運用システムの振る舞いを観察することで，運用システム開発者が未知の問題を発見し対処する．この試みを継続して行うことで運用システムの耐障害性を向上させることができる．

これらの FIT ツールはインスタンスの強制停止のように瞬間的に発生する障害や一定確率でのパケットロスといった障害注入シナリオを記述することで障害を注入する．しかし，これらの障害注入シナリオでは CPU 使用率やディスク使用率の変動など時系列的な変化によって発生する障害を注入することはできない．そのため，時系列的な状態変化を異常として検出するシステムの実稼働環境における動作検証や挙動確認のためのテストには適用できない．

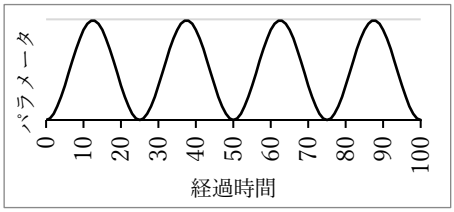
3 提案システムの設計

3.1 提案システムの設計方針

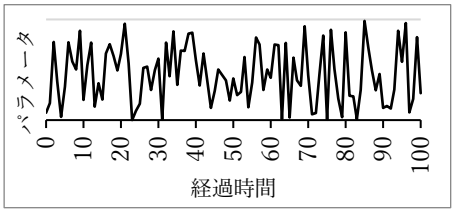
提案システムでは障害につながる可能性のある運用システムの時系列的な状態変化を障害サインと呼ぶ．具体的には CPU 使用率やディスク使用率の変動など，システムの状態を示す個々の値やその組み合わせが時間的に変化する時系列データのことを指す．障害サインの例を図 1 に示す．



(a) 線形的な変化



(b) 周期的な変化



(c) 不規則（ランダム）な変化

図 1 提案システムで表現する障害サインの例

運用システム開発者が意図する障害サインを表現できるようにするには, 表現する障害サインの内容を具体的に指定できる方式が必要となる．また既存の障害注入ツールを用いて障害サインの表現を可能とするには，障害注入ツールに使うパラメータを時間経過に伴って自動的に更新する制御機構も必要である．更に Chaos Engineering の原則[7]にもあるように，提案する FIT システムは検証による被害を最小限に抑えられる構成にすることが望ましい．そこで，提案システムでは以下の 3 点を考慮した設計にする．

- 障害サインを表現するために注入する障害についての 6 つの項目 (障害注入対象, 障害の種類, 障害の挙動, 障害の挙動の詳細, 障害発生確率, 障害注入期間）を構造化した障害注入シナリオから障害注入が行えること．
- 障害注入シナリオの内容に応じて障害注入を自動的に制御する機構を設けること．
- 障害注入対象側で障害注入によって発生した必要以上の障害の発生を検出でき，かつ検出した場合にただちに障害注入を終了できる構造にすること．

3.2 提案システムの構成

図 2 に提案する FIT システムの構成を示す．運用システム開発者はあらかじめエージェントベースの監視ツールによりサービス提供サーバのデータを収集，蓄積および可視化し，現在の状態を詳細に把握しておく．まず，現在の状態をもとに運用システム開発者は図 3 のようなフォーマットで障害注入シナリオを作成し，障害注入管理部に投入する．障害注入シナリオは障害注入管理部を通して障害注入シナリオに沿った内容の障害注入制御パラメータに変換され，障害注入通知部から障害注入制御部に送信される．障害注入制御部は受信した障害注入制御パラメータに応じて障害注入ツールを用いた障害注入を行い，障害サインを表現する．なお，障害注入を行っている間，障害注入の実行状況を表すメッセージを障害注入制御部から障害注入通知部に送信し障害注管理部に通知することで，障害注入の実行状況を更新する．

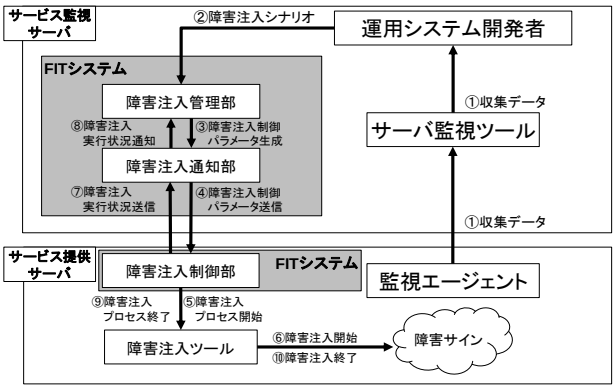


図 2 提案システムによる障害注入の流れ

```
scenario:                                          # シナリオに関するメタデータ
   scenario_name: example scenario                   # シナリオ名
   scenario_desc: Packet delay to multiple regions.  # シナリオに関する説明文
   scenario_id: 1 (FITシステム内で生成・付与)          # シナリオのID
   creation_date: 2019-10-16 13:00:00 (FITシステム名で生成・付与) # シナリオ生成・投入時刻

failures:                                          # 障害（障害注入プロセス）に関するデータ
  - failure_name: network failure                  # 障害名
    failure_desc: Packet_delay will increase gradually.  # 障害に関する説明文
    failure_id: 1 (FITシステム内で生成・付与)        # 障害ID
    parameters:                                    # 障害に関するパラメータ
       hosts:                                        # 障害注入対象
         - 192.0.2.10
         - 192.0.2.11
       type: p_out_delay                             # 障害の種類（cpu, memory, hdd, p_out/in_delay, ...）
       behavior: cycle                               # 障害の挙動（linear, cycle, random,...）
       rules:                                        # 障害の挙動の詳細
          range: 100ms                                  # 変化させる範囲
          interval: 1h                                  # 周期
       percentage: 100%                              # 障害発生確率
       duration:                                     # 障害注入期間
          begin: 2019-10-17 09:30:00                    # 障害注入開始時刻
          end: 2019-10-17 18:30:00                      # 障害注入終了時刻

  - failure_name: ...
```

図 3 障害注入シナリオのフォーマットおよび記述例

障害注入を途中で強制終了する場合は，障害注入を強制終了する本システム専用のコマンドを障害注入管理部で実行することで，障害注入制御パラメータを生成および送信し障害注入を強制終了する．

障害注入制御部から障害注入通知部へのメッセージの送信に連続で失敗した場合は，障害注入制御部側で必要以上の障害が発生したと判断し独立的に障害注入を強制終了する．

4 まとめと今後の展望

運用システムに対して既存のFITツールではできない，時系列的な状態変化の表現ができるFITシステムの概要を示した．

今回の提案では主に時系列変化のある障害を表現できるようにすることに焦点を当てているが，実稼働環境においては既存のFITツールで実現できる突然のサーバダウンのようにバーストな障害による周囲への余波を異常検出に利用することも想定される．そのため，将来的には時系列変化のある障害だけでなく，バーストな障害も合わせて注入できるFITシステムを開発していきたいと考えている．

謝辞

本研究の一部はJSPS科研費19K11929, 18K11266とMIC/SCOPE（162108102）の支援を受けて実施しています．

参考文献

[1] T. Kimura, A. Watanabe, T. Toyono, K. Ishibashi, “Proactive failure detection learning generation patterns of large-scale network logs,” 11th IEEE International Conference on Network and Service Management, CNSM 2015, pp. 8-14, Nov. 9-13, 2015.

[2] K. Yammual, P. Phunchongharn, T. Achalakul, “Failure Detection through Monitoring of the Scientific Distributed System,” IEEE International Conference on Applied System Innovation, ICASI 2017, pp. 568-571, May. 13-17, 2017.

[3] K. Andrus, N. Gopalani, B. Schmaus, “FIT: Failure Injection Testing,” The Netflix Tech Blog, https://medium.com/netflixtechblog/fit-failure-injection-testing-35d8e2a9bb2, Oct. 23, 2014.

[4] A. Basiri, N. Behnam, R. de Rooij, L. Hochstein, L. Kosewski, J. Reynolds, C. Rosenthal, “Chaos Engineering,” IEEE Software, Vol. 33, No. 3, pp. 3541, May-June, 2016.

[5] alexei-led/pumba, https://github.com/alexei-led/pumba (2019/10/20参照)

[6] Netflix/chaosmonkey, https://github.com/netflix/chaosmonkey (2019/10/20参照)

[7] “Principles of Chaos Engineering,” http://principlesofchaos.org/?lang=ENcontent (2019/10/20参照)