

# Detecting Memory Leaks through Introspective Dynamic Behavior Modeling using Machine Learning

*ICSE'14*, May 31 – June 7, 2014, Hyderabad, India

Sangho Lee, et al.

---

2017/03/21

# 論文の概要

---

- ・メモリリークの判定にstalenessを使用
  - Staleness: オブジェクトがどれだけ解放されず放置されているかの指標(後述)
  - 他にも色々な指標を用意
- ・評価(precision, recall, accuracy)
  - 人工リークプログラムの検出
  - 現実にリークが生じているプログラムのリーク箇所の検出
  - →手動(マニュアル)より高い精度を記録
- ・フューチャーワーク
  - 今回使わなかった指標を組み込むことで、更に高精度にできるのでは

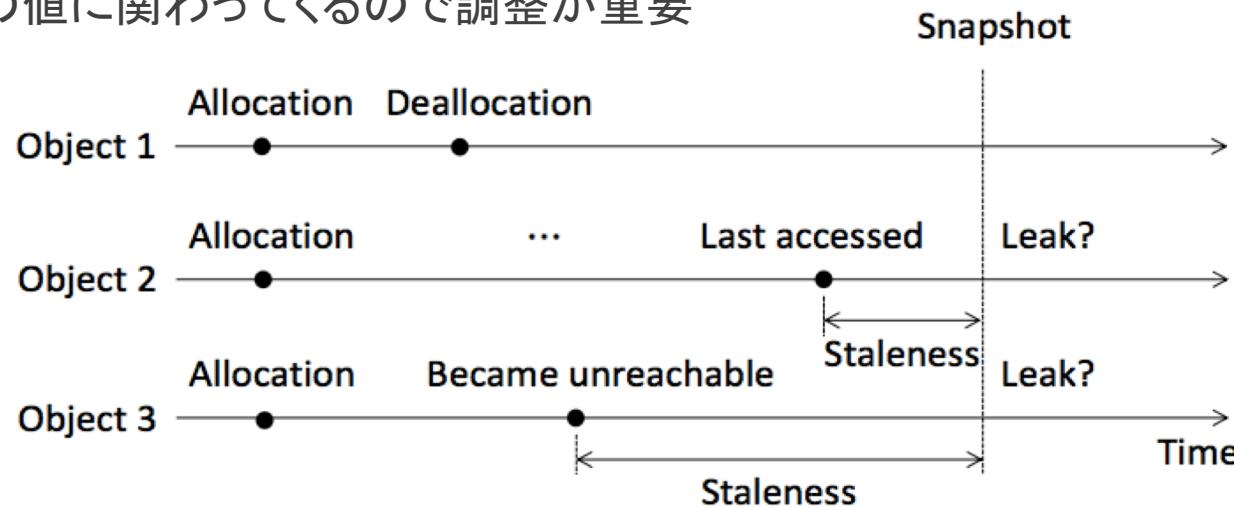
# そもそも

---

- ・メモリリーク: 割り当てられたメモリが正しく解放されずにメモリを食いつぶす現象
  - 食いつぶしによってパフォーマンスの低下やアプリのクラッシュなどが生じる
  - 判定は基本的に難しい
- ・機械学習: 教師有り学習と教師無し学習に大別できる
  - 今回のフレームワークは教師有り学習
    - →サンプルの質が大事

# 提案手法の前提

- staleness: スナップの段階で、開放されていないオブジェクトが最終アクセスからどれだけ時間が経っているかの指標
  - →直感的に、リークを生じているオブジェクトは最終アクセスから時間が経っているはず
  - →stalenessが高い=リークを生じている となるのでは?というアイデア
- ヒープスナップショット(スナップ): stalenessを計測するためにオブジェクトをチェックすること
  - →直接stalenessの値に関わってくるので調整が重要

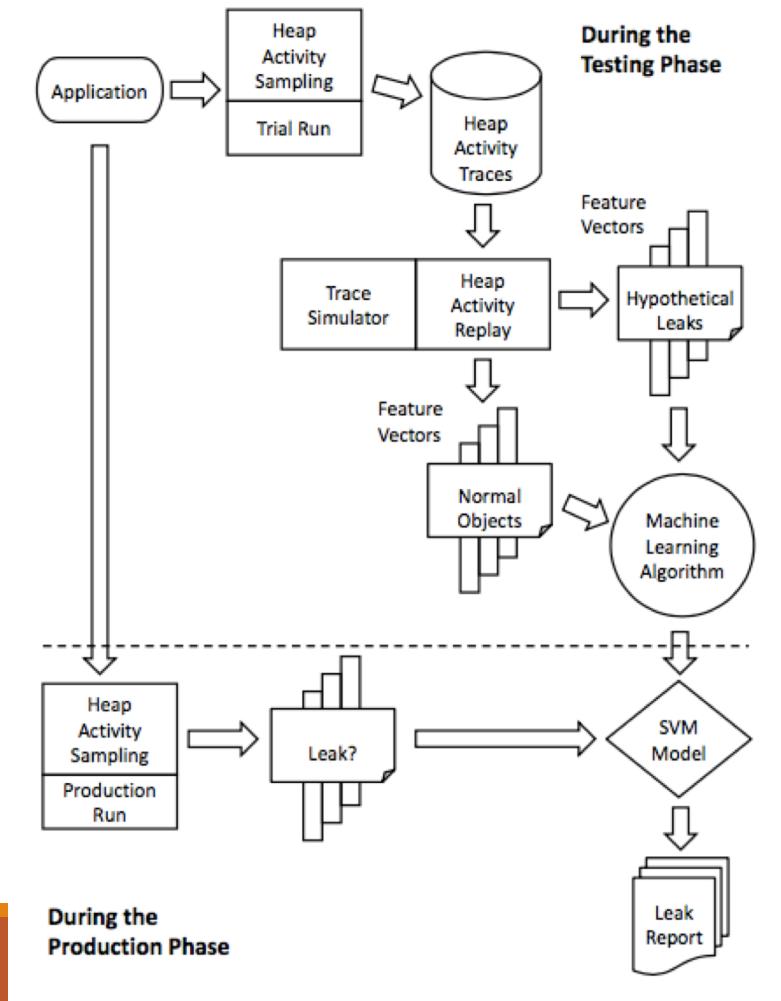


# 提案手法の前提

---

- stalenessの補足
  - 類似オブジェクトは基本的に寿命が近い
  - →類似オブジェクトの中で、異様に寿命が長い(stalenessが高い)モノはリークされている可能性大
- 教師有り機械学習にあたり、訓練例(training examples)が必要
  - この訓練例に、非リークオブジェクトの例、リークオブジェクトの例が含まれる
  - →矛盾があるのが分かりますか
    - →リーク判定を機械にしてもらうのに、機械はリーク判定の例が必要...
    - →これの解消にwhat-if-leak-stalenessという指標を用意(後述)
- この提案手法は基本的にアプリのテスト段階でのリークの検出を補助する目的
  - 人の目には映りづらい潜在的リークの検出

# 提案フレームワークによる検出の流れ



## ・大きく分けて2つのフェーズ

- テストフェーズ: 訓練例を作成する。機械学習をする。
- 製品フェーズ: 実際に実行してリークを判定
- 各フェーズ1回、計2回のアプリ実行

## ・テスト実行

- ↓
- 機械学習＆リークパターンをモデリング
- ↓
- 製品実行＆ヒープメモリの挙動をロギング
- ↓
- モデルと照らし合わせてリークを検出

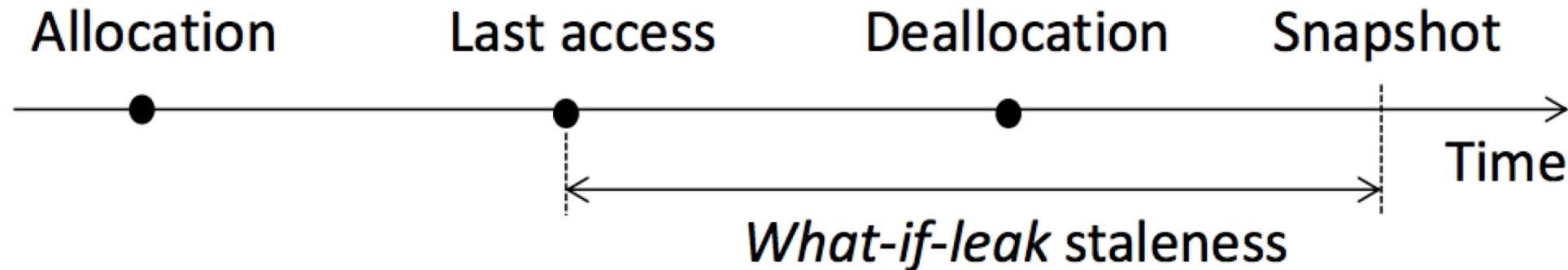
# 既存手法との差異

---

- ・これまでstalenessを使ったリーク検出技術は存在
  - 既存手法ではstalenessを評価する閾値を人間側が設定していた
    - →しかしオブジェクトによって適切な閾値は異なる
    - →それを人間側が正確に設定するのは限りなく難しい
  - →なら機械に学習してもらって勝手にやってもらおう(モチベ)

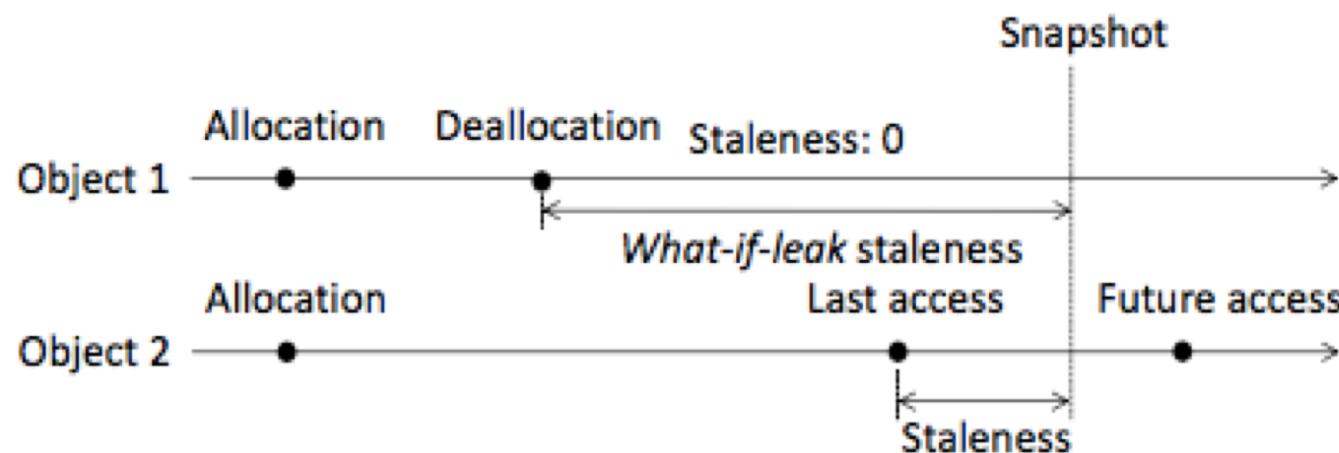
# what-if-leak-staleness

- ・**解放済み**オブジェクトの最終アクセスからスナップまでの時間
  - what-if-leak-staleness < staleness となるオブジェクトがあれば、それは仮定リークとみなす
  - この仮定リークを基準にリーク例と非リーク例を生成
    - →精度は？ →評価を見る限り充分な性能を保持



# 訓練例の出力

- ・訓練例出力の例(下図: Obj1,Obj2は類似オブジェクトという前提)
  - Obj1: 解放済みという例 & what-if-leak-stalenessによる仮定リークの例
    - →計2例 (what-if-leak-stalenessはこの場合AllocationからSnapまでじゃないの?)
  - Obj2: Obj1から得られたwhat-if-leak-stalenessと比較して、Obj2が非リークの可能性があるという例
    - →what-if-leak-staleness > staleness だからリークとは言い切れない(恐らく非リークの1例として処理)
    - →計1例



# リークの判定

- SVM(support vector machine): 教師あり学習の認識モデルの一種
  - リークかどうかの判定は二つの仕事
  - 先の訓練例の出力は、このSVMにを作成するため
- 製品実行時に各オブジェクトoから得られる情報
  - T<sub>allocation</sub>(o): oが割当てられた時刻
  - T<sub>snapshot</sub>: スナップを撮った時刻
  - T<sub>last\_accessed</sub>(o): oへの最終アクセス時刻
- ↑これら情報を正規化(Normalize)
  - 評価の値を0~1に収めることで精度を高める
  - NT<sub>allocation</sub>(o): 正規化した割り当て時刻
  - NT<sub>staleness</sub>(o): 正規化したstaleness

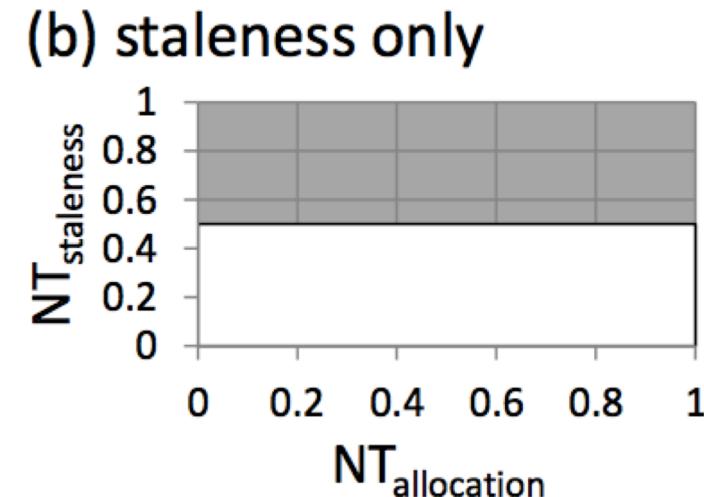
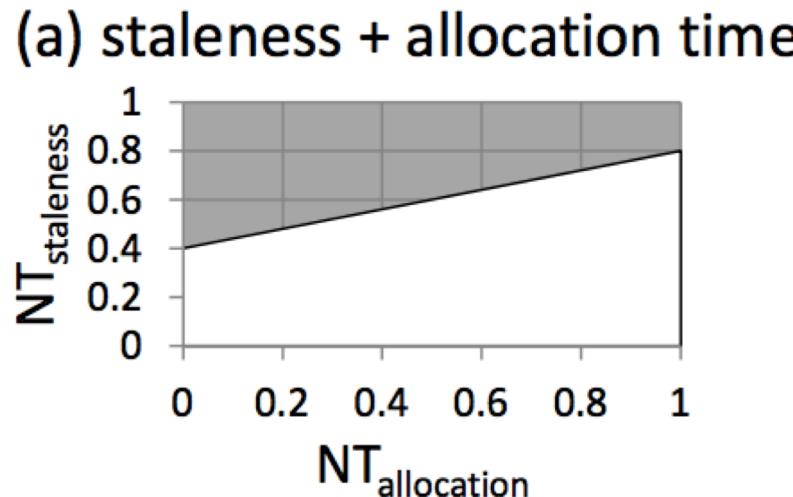
$$NT_{allocation}(o) = \frac{T_{allocation}(o)}{T_{snapshot}}$$

$$NT_{staleness}(o) = \begin{cases} 0, & \text{if } o \text{ is deallocated prior to } T_{snapshot} \\ \frac{T_{snapshot} - T_{last\_accessed}(o)}{T_{snapshot} - T_{allocation}(o)}, & \text{otherwise} \end{cases}$$

where  $T_{allocation}(o)$  represents the allocation time of  $o$ , and  $T_{last\_accessed}(o)$  represents the last access time of  $o$ <sup>2</sup>.

# リークの判定

- SVMは例えば下図のような形で表現される
  - これは訓練例を元に分布化される
  - 製品実行時に得られた情報( $NT_{allo}$ ,  $NT_{stale}$ )をこのSVMに適応
    - 白枠内にいれば非リークと判定
    - そうでなければリークと判定



# スナップの決定

---

- ・あくまでスナップは人が設定するが、これは特定が容易
  - →このフレームワークの対象が、Webサーバに代表される長時間特定の挙動を繰り返すアプリだから
  - ここは人の目なんやね...
    - →著者視点では労力は スナップの決定(提案) < stalenessの決定(既存) の模様
- ・ただし、リークの判定に大きく響いてくる要素だけに、慎重にスケーリングをして精度を保持
  - 先の正規化と同様の考え方
  - $T_{train\_snapshot}$ : テスト実行時の時刻
  - 製品実行時は、この $T_{train\_snapshot}$ を元に補正
    - →製品実行時の時刻が遅い程補正が強くなる

$$r = \frac{T_{snapshot}}{T_{train\_snapshot}}$$

$$TS(t) = T_{train\_snapshot} * \left(\frac{t}{T_{snapshot}}\right)^r$$

# その他の指標

---

- ・先のNTallocation(o), NTstaleness(o)の他に、より精度を高めるための指標を用意
  - →指標が多いほど正確性が増す
  - →先程のSVMが3次元、4次元的なマッピングに成り得る

- ・NSize(o):

- oのサイズが類似Objの最大値と比較しての割合

$$NSize(o) = \frac{Size(o)}{\max_{obj \in objects} Size(obj)}$$

- ・NSite(o):

- oがそもそもどこに割り当てられたか
    - →同じ場所に割り当てられるオブジェクトの寿命は近いから?

Encoding allocation site, on the other hand, uses a vector of binary variables. Formally, the vector  $NSite(o)$  is defined as,

$$NSite(o) = (Site_1(o), Site_2(o), \dots, Site_{n-1}(o), Site_n(o))$$

$$Site_k(o) = \begin{cases} 1 & \text{if } o \text{ is allocated from allocation site } k \\ 0 & \text{otherwise} \end{cases}$$

# 評価

---

- ・リーク検出の基本的評価指標

- Precision(精度)
  - リークと判定際の正解率
- Recall(再現率)
  - 実際のリークをどれだけ検出できたか

- ・分類別精度

- Classification accuracy(accuracy)
  - 交差検証にて使う指標
    - 交差検証: 解析対象標本を分割(例えばAとB)して、
      - Aを先に解析し、Bでテストをする。解析の精度を評価する近似手法
    - →オブジェクト別の精度を求める上でも評価指標として適している

$$\text{accuracy} = \frac{|\text{correctly identified objects}|}{|\text{objects}|}$$

$$\text{precision} = \frac{|\text{reported\_leaks} \cap \text{real\_leaks}|}{|\text{reported\_leaks}|}$$

$$\text{recall} = \frac{|\text{reported\_leaks} \cap \text{real\_leaks}|}{|\text{real\_leaks}|}$$

# 評価

- 今回の比較は、3本を比較
  - Manual(既存)
  - Base-model(比較用の機能制限モデル)
  - Full-model(今回の提案手法)

