OpenPNEのソフトウェアエージングに関する研究

東京都立大学　*近藤和希　KONDO Kazuki
会員　東京都立大学　肖霄　XIAO Xiao

1. 序論

デジタル技術が発展した現代では，私たちの身の回りから宇宙規模のものまでがソフトウェアシステムで構築されており，非常に長時間の連続稼働が必要なものも数多く存在する．一方で，開発時のソフトウェアテストで得られるバグは，明確に定義された条件下のものであり，リリース前に完全に原因を解決することは難しい．このような連続稼働による予期しない性能劣化が生じる現象は「ソフトウェアエージング (老化)」と呼ばれる．現代では，特に高可用性が要求される最新のソフトウェアシステムに対する要求が高まり，それらのソフトウェアエージングのリスクを定量的に評価する必要性が高まっている．ソフトウェアエージングのリスクが観測された場合，次に検討すべきはソフトウェア若化手法の検討である．ソフトウェアエージング関連のバグが蓄積し，それがシステムの性能劣化として現れることを未然に防ぐことが目的である．

本研究では，ソフトウェアエージングの研究対象として，SNS システムの一つである OpenPNE を選択する．SNS ならではの変動する負荷期間を想定し，監視期間における 3 種類のメトリクスに着目する．その結果，比較的低い一定負荷においても，少なくともソフトウェアエージングの潜在可能性を確認できた (暫定)．

2. 関連研究と動機

2.1. 関連研究

1984 年に Adams[1] による継続的なソフトウェア稼働によるパフォーマンスの低下が報告された後，1994 年に Parnas[2] によってソフトウェアエージングという用語が最初に使用されたとき，多くの研究者は驚かされ，それ以来広範な研究が行われてきた．複数のソフトウェアエージングの定義が存在するが，本研究では，Huang ら [3] によるシステムの実行時間に焦点を当てたプロセスエージングと呼ばれるソフトウェアエージングの定義に基づき考える．

Dohi ら [4] によると，ソフトウェアエージングの研究は，Threshold-based Approach，Machine learning-based Approach，Measurement-based Approach の 3 つの手法に分類できる．本研究では特に，Measurement-based Approach を用いた研究について調査し，多種多様なシステムを対象に行われていることが分かった．以下，書き方 1 と 2 のどちらかで検討中．1. 例として，IBM のクラウドシステム [?] や画像判定アプリケーションをクラウド環境やエッジ環境にデプロイしたもの [?][?][?]，さらに Android OS[?][?][?] やブロックチェーンにまで至る．2. 本分野の研究は，Web アプリケーション [?][?] やクラウドシステム [?][?][?][?] といった多種多様なシステムを対象にし，それぞれのソフトウェアエージングの振舞とその原因を調査している．

2.2. 動機

本研究では，SNS システムの一つである OpenPNE[1] を対象に Measurement based Approach を用いて，ソフトウェアエージングの潜在可能性を調査する．これまでの研究の中で直接的に SNS システムを対象にしたものは，私たちの知る限りない．現代の SNS システムは拡張性も高くユーザー数も莫大であることに伴い，X 社のような運営企業にとって，サーバーの運用リスクは高まっているといえる．本研究では，これまでの研究でも採用されてきた高/中/低負荷に加え，SNS システム独自の多種多様な負荷シナリオを検討し，それに応じたソフトウェアエージングの調査を行う．また，本研究では，Torquato ら [5] が 2018 年に，ストレス期間，待期期間，若化期間のそれぞれでリソースの監視を行った研究を参考にし，本研究では監視期間というものを考える．ただし，本研究では若化期間には踏み込まない．

3. 本研究の実験環境 (予備知識)(名検討中)

3.0.1. 環境設定

※図の記載については検討中．この環境では，Docker により，サーバー PC の OS と OpenPNE の運用環境を隔離している．また，OpenPNE の環境を Docker 上に独立させることでメトリクスをよりクリーンに取得でき，移植性も高い．一方で，負荷ツールとの間を LAN ケーブルでつなぐため，物理的な接続の影響を受けるリスクがある．

物理デバイス:

- サーバー:HP Z2 mini G9 workstation desktop PC，13th Gen Intel(R)Core(TM) i9-13900 3.00GHz，RAM:16.00GB，Win11 pro，23H2

[1] https://www.openpne.jp/about/

- クライアント:Desktop-7CL98lG, Intel(R) Core(TM) i7-6600U CPU 2.60GHz, RAM:8.00GB, Win10 Enterprise, 22H2
- LAN ケーブル

データ収集:

- サーバー側:Docker stats コマンドで取得 (CPU 使用率, メモリ使用量)
- クライアント側:Jmeter(リクエストエラー率※以下エラー率と表記)

4. 実験

4.1. シナリオ 1: 一定負荷

本節では, 一定負荷によるエージングの影響の調査を行うが, 本環境のサーバー性能調査も兼ねている. 現実の状況において, 定常状態に相当すると考えている.

表 1: エラー率と推定 RPS:error rate and estimated RPS(有効数字 2 桁)

RPS	error rate[%]	estimated RPS	Risk of SA
10	0.87	-	○
30	56.51	13.5	×
50	72.75	13.63	○
70	80.20	15.33	×
Avg	-	14.15	-

表 1 は, 複数の一定負荷シナリオのエラー率, およびそれらの値から推定できる RPS を示したものである. この結果から, 本環境の耐久 RPS は 14.15RPS であると推定する. また, メモリ使用量については, Mann-Kendall 検定と Sen の傾き推定の結果から, 少なくとも 10RPS と 50RPS でソフトウェアエージングのリスクが認められた.

4.2. シナリオ 2: 増加負荷 (監視期間 2H)

4.1 の結果より, 1〜15RPS の増加変動を行い, バズ現象を想定する. 対照実験として, 5, 10, 15RPS の一定負荷と比較し, 監視期間における各メトリクスの変動に着目し, ソフトウェアエージングの影響の差異に着目する.

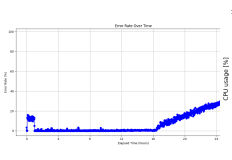
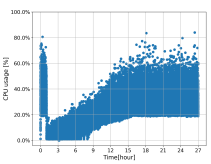


図 1: エラー率



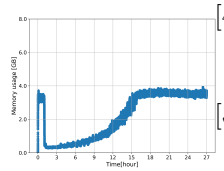


図 2: CPU 使用率 図 3: メモリ使用量

図 1, 2, 3 は, 1〜15RPS の増加負荷における各メトリクスの時系列変化である. スレッド立ち上げ期間は負荷の大きさから各メトリクスが比較的高い値を示している. 増加負荷の期間については, 増加に伴い, 全メトリクスが増加していることが分かる. 特に, エラー率については, RPS が 10RPS 程度に至る 15 時間ごろから極端に増加しており, 推定 RPS よりも低い負荷の段階でエラーを生じた. 一方で, 他二つのメトリクスについては, その時間帯までは増加傾向にあったが, それ以降は極端な増加は少なくとも見られない. これがソフトウェアエージングによる影響かどうかは現時点では判断できない. また, 着目したいのは, 負荷後 2 時間のメトリクスの推移である. 以降については, 追加実験の結果が出てから.

5. 結論と今後の展望

結論については結果がそろってから.

本研究では, 比較的シンプルな SNS の設定で行ったが, より高度な拡張機能を有したものにすべきといえる. 近年では, 序論や関連研究でも述べたように SNS だけでなく, 多くの Web サーバーは, クラウド環境に用意されている. そのため, 今後 SNS をクラウド環境にデプロイするなどして, 大規模でより現実な負荷テストを行うことが望ましいと言える. また, 本研究ではソフトウェア若化手法についての検討を行っていない. 再起動やガベージコレクションの管理などの代表的な若化手法を, いつ実行すべきかを検討する必要がある.

参考文献

[1] Adams, N.Edward, “Optimizing Preventive Service of Software Products,” IBM Journal of Research and Development, vol. 28, no. 1, pp. 2-14, 1984.

[2] D.L.Parnas, “Proceedings of the 16th International Conference on Software Engineering (ICSE1994),” Software aging, pp. 279-287, 1994.

[3] Y.Huang, C.Kintala, N.Kolettis and N.D.Fulton, “Software Rejuvenation: Analysis, Module and Applications,” 25th International Symposium on Fault-Tolerant Computing(FTCS1995), pp. 381-390, 1995.

[4] T.Dohi, K.Trivedi and A.Avritzer, “Handbook of Software Aging and Rejuvenation,” WORLD SCIENTIFIC, pp. 73-90, 2020.

[5] M.Torquato, Araujo, Matheus, Jean, I.M.Umesh, and P.Maciel, “SWARE: A Methodology for Software Aging and Rejuvenation Experiments,” Journal of Information Systems Engineering & Management vol. 3, 2018.