継続的インテグレーションを用いたLinuxカーネルの性能劣化検知システムの提案と実装

馬場 祐希[1,a)]　宮島 敬明[1,b)]

概要：システムソフトウェアの性能劣化は発見が遅れるほど原因の特定に手間がかかるため，コミットごとに検知することが重要である．しかし，多数のベンチマークを毎回すべて実行するのは計算資源の限られた環境では難しく，既存の回帰テスト選択は性能劣化を検知するベンチマークの選択には直接適用しにくい．本研究は，Linux カーネルを対象に，コミットに応じて実行するテストを絞り込み，少ない資源で性能劣化を検知することを目的とする．本稿では，コミットの差分とメッセージから大規模言語モデル（LLM）がテストを選び，継続的インテグレーション（CI）を用いて実機で実行する手法を提案し，GitLab CI で試作した．送信するパケットの順序と時刻を制御する Fair Queue（FQ）スケジューラで過去に起きた性能劣化を題材に，確かめる範囲の広さ（粒度）が異なる 4 種類のテストを作成して評価した．通信全体を測る粗い粒度のテストは劣化に至る前の破損を検知できず，LLM はテストの説明だけから検知に必要なテストを選んだが，対応するテストのない変更に無関係なテストを選ぶ誤りも 1 件あった．これは，特定の処理を直接確かめる細かい粒度のテストが必要であることと，LLM がテストの選択を担えると考えられる．

1. 研究背景：性能劣化と検出手法

Linux カーネルに代表されるシステムソフトウェアの多くは非常に大規模なコードベースから成り，多数の開発者によって絶えず変更されている．そのため，ある変更が別の箇所の性能劣化を引き起こすことが知られている [1]．性能劣化は機能的なバグと異なり機能テストの失敗として現れにくく，変更箇所から離れて現れることがあり，リリース後に発見されることが多い．その原因特定には正常な版と劣化した版の間のコミットを二分探索するが，それぞれにビルド，再起動，ベンチマークが必要で手間が大きい．また，リリース候補や多数の開発ツリーに対してベンチマークや機能テストを広く実行する既存の取り組みもあるが [1], [2]，ベンチマークは種類が多く，計算資源の限られた環境で毎回すべてを実行するのは難しい．この他，ソフトウェア工学の分野では変更に関係するテストだけを選ぶ回帰テスト選択が研究されている [3]．これはテストとファイルの依存関係に基づいて機能テストを選ぶもだが，変更から離れた箇所に現れる性能劣化を検出するベンチマークの選択には適用しにくい．本研究は，コミットが入った時点で性能劣化を検知し，混入を未然に防ぐことを目指す．

[1] 明治大学 理工学部 情報科学科
〒 214-0033 神奈川県川崎市多摩区東三田 1-1-1
[a)] ce265044@meiji.ac.jp
[b)] takaaki.miyajima@cs.meiji.ac.jp

2. 提案手法と試作システム

本研究では，コミットの差分とメッセージを大規模言語モデル（LLM）に読ませて関係すると考えられるベンチマークだけを選び，実機で実行して性能劣化を知らせる手法を提案する．ここでテストとは，ベンチマーキングと，その結果に対する合格の条件の組を指す．

2.1 提案手法

図 1 に提案手法の処理の流れを示す．最終的には, Linux のアップストリームのコミットの監視を想定しているが，本稿の試作システムでは，開発者がカーネルへの変更を `git format-patch` の形式で GitLab のリポジトリに追加すると，CI パイプラインが起動する．ジョブは中間マシン上の GitLab Runner が受け取り，SSH 経由でテスト機を操作する．テスト機は測定のたびに再起動するため，Runner をテスト機の外に置くことで，再起動をまたいでジョブを継続できるようにした．

1 回の CI パイプライン実行では，まず LLM に差分，メッセージ，テストのカタログを渡してベンチマークを選ばせ，何も選ばれなければビルドせずに終える．何かしらのテストが選ばれた場合は，カーネルをビルドして再起動し，選ばれたテストだけを実行して FAIL があればジョブを失敗させ，最後に正常なカーネルに戻す．結果は JUnit 形式で

© 1959 Information Processing Society of Japan

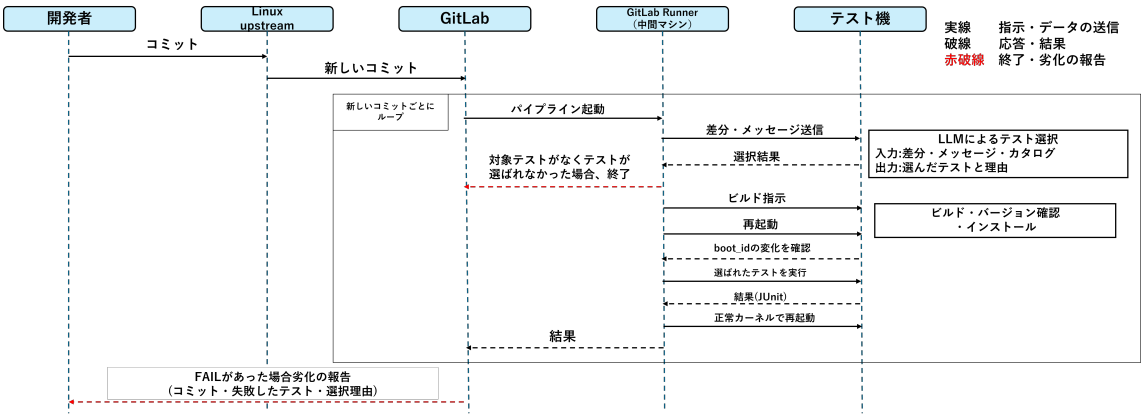


図 1　継続的インテグレーションと LLM を用いた性能劣化検知手法の処理の流れ

GitLab に送り，失敗したテストと LLM が選んだ理由を画面で確認できる．なお，LLM が誤って「関係なし」と判断すると劣化を見逃すため，一定期間ごとに全テストを実行するなどの安全策を併用する．

2.2　LLM によるベンチマーク選択

スループットを測るようなベンチマークは特定のファイルや関数に対応づけにくく，その一覧も保守し続ける必要がある．そこで，差分とメッセージから変更の意図を読み取り，テストの説明と対応づけられる LLM を用いる．各テストの説明，粒度，所要時間を JSON のカタログに記述し，ファイル名や関数名は与えない．LLM には「結果が変わりうるテストだけをなるべく狭く選ぶ」「無関係なら選ばない」と指示し，選んだテストと理由を JSON で返させる．LLM に Claude Haiku 5.5 を用い，呼び出し部分は他の LLM に差し替えられるようにした．

2.3　テストケースと初期の結果

提案手法を検証するテストケースとして，転送パケットに残った受信時刻を FQ スケジューラが送信予定時刻と誤解し，パケットを破棄した 2018 年の不具合（コミット番号：`8203e2d844d3`）を用いた．修正後のカーネルには，(i) 転送時に時刻を消す処理と，その後加わった (ii) FQ が時刻を換算する処理の二重の防御がある．1 台のマシン内に，独立したネットワーク環境を作る netns（network namespace）を 3 つ用意し，仮想的な LAN ケーブルにあたる veth（virtual Ethernet）で接続した．このテストケースで，表 1 に示す粒度の異なる 4 つの状況を作成し，LLM が変更に合うベンチマークを選べるかを調べた．防御を片方だけ外すと A または B だけが FAIL し，C と D は PASS した（両方外すと C，D も FAIL）．片方が外れても劣化はまだ表面化しないが，ミクロなテストでは劣化に至る前の破損を検知できる．変更 7 件を LLM に 3 回ずつ与えると，正解のある 4 件では全ての試行で正解を含むテストを選び（テスト実行時間は平均 35 秒，全テストで 80 秒），無関係な 2 件では選ばなかったが，対応するテストのない IPv6 の変更では誤選択した（3/3）．ただし，正解に加えて C も選ぶ傾向があった．関数一覧との照合でも同じ正解を選べるが，LLM は説明だけから選んだ．また，GitLab から防御 (i) を外すパッチを投入すると，B が選ばれて FAIL し，約 3 分で破損を検知できた．

表 1　粒度別のテスト

テスト	粒度	内容と合格条件	秒
A	ミクロ	CLOCK_REALTIME の送信時刻付き UDP を fq に入れ，換算されて届くか（防御 (ii)）	10
B	ミクロ	転送後のパケットに受信時刻が残っていないかを eBPF で数える（防御 (i)）	15
C	メソ	受信時刻の記録を有効にして ping と UDP を転送し，fq で破棄されないか	20
D	マクロ	外部ホストを模擬した TCP の転送スループットが半分未満に落ちないか	35

3.　今後の展望

本稿の評価はテストケース 1 件，選択の事例 7 件と小規模であり，関数一覧との照合でも選べる事例にとどまる．今後は，より多様な変更を用いて，LLM によるテスト選択の評価を詰めていく．あわせて，メモリなど別のジャンルの事例と既存のベンチマークを取り込んでテストを拡充し，upstream のコミットの継続的な監視と，過去の性能劣化を用いた大規模な評価を行う．

謝辞　本研究は，産業技術総合研究所の深井 貴明氏（当時）との共同研究により行われた．多大なるご協力とご助言をいただいたことに，心より感謝申し上げます．

参考文献

[1] Chen, T., Ananiev, L. I. and Tikhonov, A. V.: Keeping Kernel Performance from Regressions, *Proc. Linux Symposium*, Vol. 1, pp. 93–102 (2007).
[2] KernelCI: KernelCI, The Linux Foundation, available from `https://kernelci.org/` (accessed 2026-10-28).
[3] Gligoric, M., Eloussi, L. and Marinov, D.: Practical Regression Test Selection with Dynamic File Dependencies, *Proc. ISSTA 2015*, pp. 211–222 (2015).

© 1959 Information Processing Society of Japan