24

最近の Claude Code 品質報告に関する最新情報

— An update on recent Claude Code quality reports —

公開日	2026-04-23
原題	An update on recent Claude Code quality reports
著者	Anthropic Engineering Team
原文	https://www.anthropic.com/engineering/april-23-postmortem
翻訳	Claude（機械翻訳／Anthropic）
編集	2026-04-24

最近の Claude Code 品質報告に関する最新情報

ここ 1 か月、一部ユーザーから Claude の応答品質が悪化したという報告を受け、調査を続けてきました。私たちはこれらの報告を、Claude Code、Claude Agent SDK、Claude Cowork に影響を与えた 3 つの別々の変更にまで辿り着きました。API は影響を受けていません。

これら 3 件の問題は、4 月 20 日（v2.1.116）時点ですべて解決済みです。

本稿では、何を発見し、何を修正し、同様の問題が再発する可能性をさらに下げるために今後どう取り組んでいくかを説明します。

私たちは品質劣化に関する報告を非常に重く受け止めています。モデルを意図的に劣化させることは決してありません。また、API と推論レイヤーが今回の影響を受けていないことは即座に確認できました。

調査の結果、次の 3 つの異なる問題が特定されました。

1. 3 月 4 日、Claude Code のデフォルト推論エフォートを `high` から `medium` に変更しました。`high` モードで UI がフリーズして見えるほど長いレイテンシに遭遇するユーザーがいたのを軽減するためです。これは誤ったトレードオフでした。「デフォルトは高い知能にして、単純なタスクについてはより低いエフォートをオプトインしたい」というユーザーからの声を受け、4 月 7 日にこの変更を差し戻しました。この問題は Sonnet 4.6 と Opus 4.6 に影響しました。
2. 3 月 26 日、1 時間以上アイドル状態だったセッションから Claude の古い思考を消去する変更をリリースしました。セッションを再開した際のレイテンシを下げるためです。しかしバグにより、1 回だけではなくセッションの残り全ターンで毎回これが起こるようになり、Claude が物忘れをして繰り返しが多いように見えていました。これは 4 月 10 日に修正しました。この問題は Sonnet 4.6 と Opus 4.6 に影響しました。
3. 4 月 16 日、冗長性を抑えるためのシステムプロンプト指示を追加しました。他のプロンプト変更と組み合わさったことでコーディング品質を損ない、4 月 20 日に差し戻しました。この問題は Sonnet 4.6、Opus 4.6、Opus 4.7 に影響しました。

それぞれの変更が別々のスケジュールで異なるトラフィックに影響したため、総合すると広範かつ一貫性のない品質劣化に見えました。3 月上旬には報告の調査を始めていましたが、当初はユーザーフィードバックの通常のばらつきと区別するのが難しく、社内の利用や評価（eval）でも最初は問題を再現できませんでした。

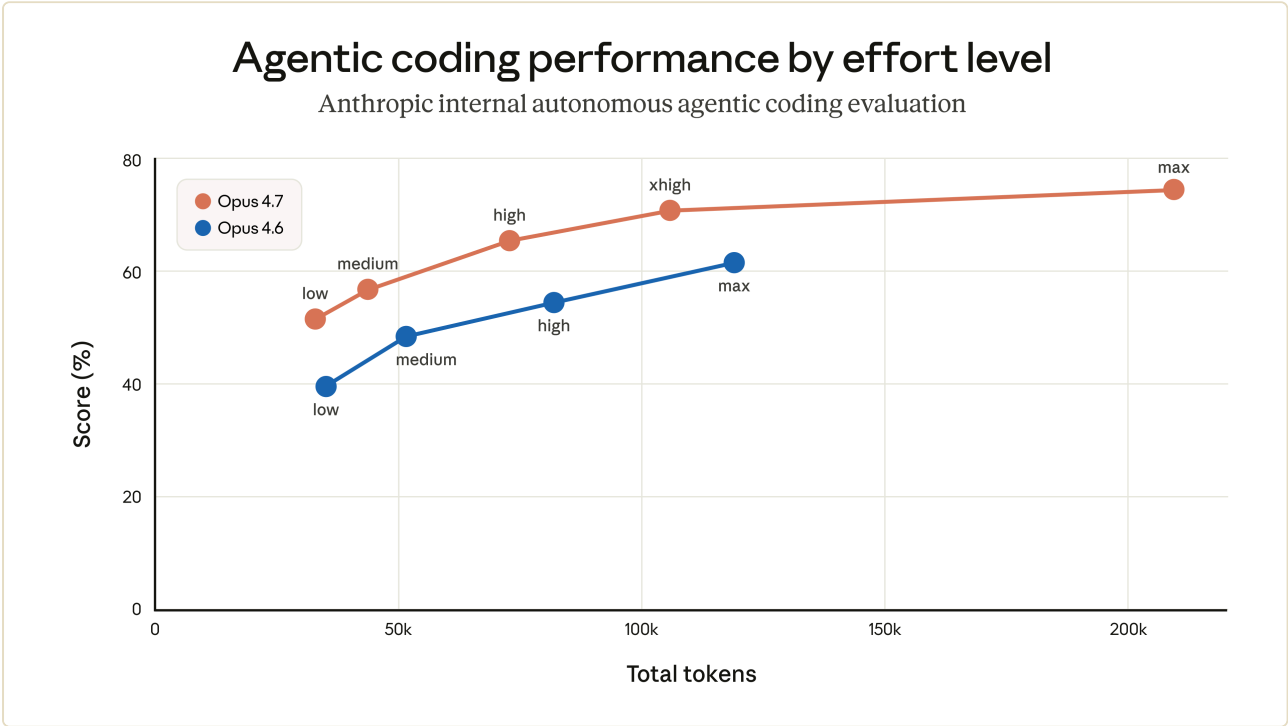
これは Claude Code で期待されるべき体験ではありません。4 月 23 日より、すべての購読者の利用上限（usage limits）をリセットします。

Claude Code のデフォルト推論エフォートの変更

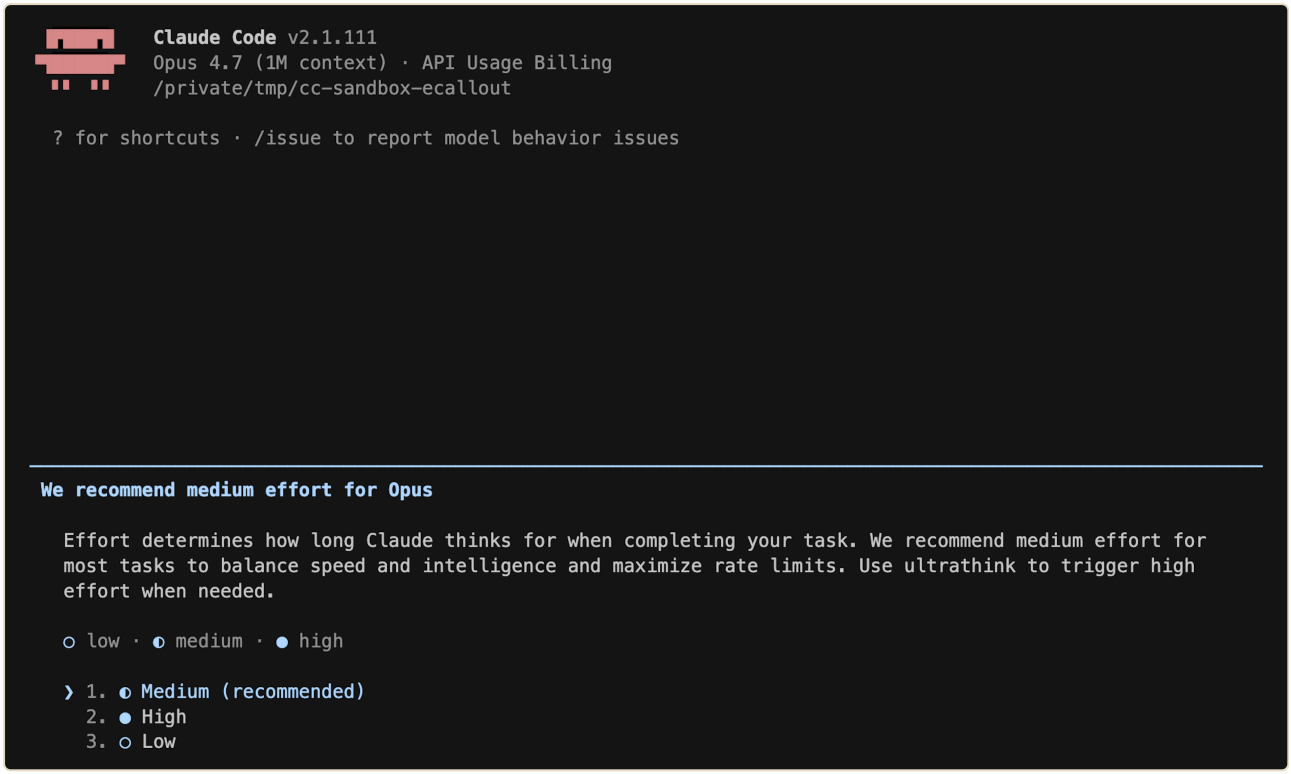
2 月に Claude Code で Opus 4.6 をリリースしたとき、デフォルトの推論エフォートを `high` に設定しました。

その後すぐに、Claude Opus 4.6 の high エフォートモードで時折思考が長すぎて UI がフリーズして見え、それらのユーザーにとって不相応なレイテンシとトークン利用が発生しているというフィードバックを受けました。

一般に、モデルが長く考えるほど出力は良くなります。エフォートレベルは、このトレードオフ——思考を増やすか、レイテンシを下げて利用上限への到達を減らすか——をユーザー自身が選べるようにする Claude Code の仕組みです。各モデルのエフォートレベルを調整するにあたって、私たちはこのトレードオフを考慮し、test-time-compute のカーブ上でユーザーにとって最も良い選択肢の幅を提供できる点を選びます。そしてプロダクト層では、このカーブ上のどの点をデフォルトにするかを決め、その値を Messages API に effort パラメータとして送ります。他の選択肢は `/effort` で切り替えられるようにしています。



社内の評価とテストでは、medium エフォートは大多数のタスクでわずかに低い知能と引き換えに、はるかに短いレイテンシを達成していました。また、時折非常に長くなる思考のテールレイテンシの問題にも悩まされず、ユーザーの利用上限を最大限活かす助けにもなっていました。その結果、medium をデフォルトエフォートにする変更をロールアウトし、プロダクト内のダイアログでその理由を説明しました。



ロールアウト直後から、ユーザーから Claude Code の知能が下がったように感じるという報告が出始めました。現在のエフォート設定をより明確にし、ユーザーがデフォルトを変えられることに気づいてもらうため、複数のデザインイテレーションをリリースしました（起動時の告知、インラインのエフォートセレクタ、ultrathink の復活など）。しかし、ほとんどのユーザーは medium エフォートのままでした。

より多くの顧客からのフィードバックを受けて、4 月 7 日にこの決定を反転させました。現在、すべてのユーザーは Opus 4.7 では `xhigh` エフォート、それ以外のすべてのモデルでは `high` エフォートがデフォルトとなっています。

過去の推論を落としてしまったキャッシュ最適化

Claude がタスクを通じて推論を行うとき、その推論は通常、会話履歴に保持されます。こうすることで、後続の各ターンで Claude は、なぜ特定の編集やツール呼び出しを行ったのかを参照できます。

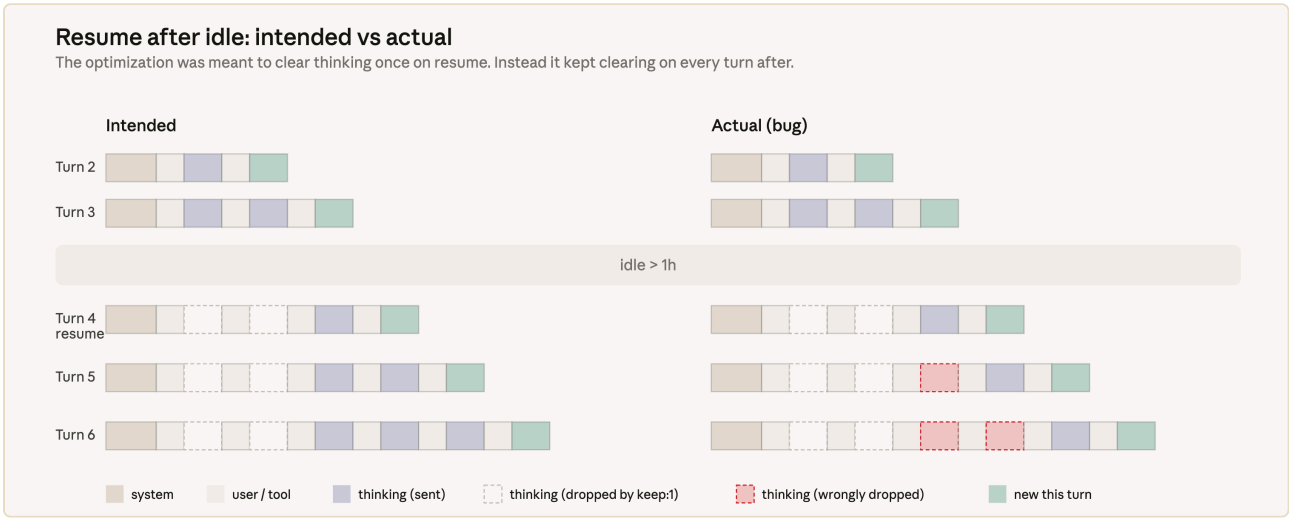
3 月 26 日、この機能の効率改善のつもりで変更をリリースしました。私たちはプロンプトキャッシュを使って、連続した API 呼び出しをユーザーにとって安く速くしています。Claude は API リクエスト時に入力トークンをキャッシュに書き込み、一定時間活動がないとプロンプトはキャッシュから追い出されて他のプロンプトのためのスペースが空きます。キャッシュ利用は慎重に管理している事項です（そのアプローチについて詳しく）。

設計自体はシンプルなはずでした。セッションが 1 時間以上アイドル状態であった場合、古い thinking セクションを消去することで、そのセッション再開時のコストを下げられます。どのみちリクエストはキャッシュミスになるため、不要なメッセージをリクエストから刈り込み、API に送る未キャッシュトークン数を減らせるわけ

です。その後は、フル推論履歴の送信に戻す。これを実現するために `clear_thinking_20251015` API ヘッダと `keep:1` を使いました。

しかし実装にバグがありました。thinking 履歴を 1 回だけ消去する代わりに、セッションの残り全ターンで毎回消去してしまっていたのです。一度セッションがアイドルしきい値を越えると、そのプロセスの残りの各リクエストで、API に対して「最新の推論ブロック 1 つだけを残してそれより前をすべて捨てる」ように指示していました。影響は累積しました。Claude がツール利用の最中にフォローアップメッセージを送ると、壊れたフラグの下で新しいターンが始まるため、現在ターンの推論すら落とされてしまったのです。Claude は実行を続けますが、自分が何をなぜ選んでいるかの記憶がどんどん失われていきました。これが、報告にあった物忘れ、繰り返し、不自然なツール選択として現れていました。

このバグは後続リクエストから thinking ブロックを継続的に落とすため、それらのリクエストはキャッシュミスにもなりました。利用上限が想定より早く消費されるという別口の報告は、これが原因だと考えています。



無関係な 2 つの実験も、最初の再現を難しくしました。メッセージキューイングに関する社内限定のサーバー側実験と、thinking の表示方法を変える直交的な変更です。後者は多くの CLI セッションでこのバグを覆い隠していたため、外部ビルドをテストしても捕まえられなかったのです。

このバグは Claude Code のコンテキスト管理、Anthropic API、拡張思考（extended thinking）の交差点にありました。導入された変更は、複数の人間および自動コードレビュー、ユニットテスト、エンドツーエンドテスト、自動検証、ドッグフーディングを通過していました。さらに、これはコーナーケース（古くなったセッション）でしか起きず、再現が難しかったことが重なり、根本原因の特定と確認に 1 週間以上を要しました。

調査の一環として、問題のプルリクエストに対して Opus 4.7 を使った Code Review をバックテストしました。完全なコンテキストを集めるのに必要なコードリポジトリを与えた場合、Opus 4.7 はバグを発見しましたが、Opus 4.6 は発見できませんでした。同様の事態を防ぐため、コードレビューのコンテキストとして追加のリポジトリをサポートするよう現在対応を進めています。

このバグは 4 月 10 日 v2.1.101 で修正しました。

冗長性を抑えるためのシステムプロンプト変更

私たちの最新モデル Claude Opus 4.7 には、前モデルと比べて特筆すべき挙動の癖があります。リリース時に書いたとおり、かなり冗長になりがちなのです。これにより難しい問題で賢くなる一方、出力トークンも増えます。

Opus 4.7 のリリースの数週間前から、準備として Claude Code のチューニングを始めました。モデルごとに挙動が少しずつ異なるため、各リリース前にハーネスとプロダクトをそのモデル向けに最適化する時間を設けています。

冗長性を下げる手段はいくつかあります。モデルトレーニング、プロンプト、プロダクトでの thinking UX 改善などです。最終的にこれらすべてを使いましたが、システムプロンプトへの 1 つの追加が、Claude Code の知能に不釣り合いに大きな影響を与えました。

> *“Length limits: keep text between tool calls to ≤25 words. Keep final responses to ≤100 words unless the task requires more detail.”*

数週間にわたる社内テストと、私たちが走らせていた評価セットで回帰が出なかったことから、この変更に自信を持ち、4 月 16 日の Opus 4.7 リリースと同時に投入しました。

この調査の一環として、より広範な評価セットを使ってアブレーション（システムプロンプトから行を取り除いて各行の影響を理解するテスト）を追加で実施しました。その評価のうちの 1 つで、Opus 4.6 と Opus 4.7 の双方で 3% の低下を確認しました。私たちは 4 月 20 日のリリースの一部として、このプロンプトを即座に差し戻しました。

今後に向けて

こうした問題を避けるために、今後いくつか取り組みを変えていきます。社内スタッフの利用で、新機能のテスト用ビルドではなく、Claude Code の公開版ビルドをそのまま使う比率を増やすこと。そして、社内で使っている Code Review ツールを改良し、その改良版を顧客にも提供することです。

システムプロンプト変更に対する統制も強化します。Claude Code のシステムプロンプト変更ごとにモデル別の広範な評価スイートを走らせ、各行の影響を理解するアブレーションを継続し、プロンプト変更をレビュー・監査しやすくする新しいツールも構築しました。加えて、モデル固有の変更が対象のモデルだけにゲーティングされるよう、CLAUDE.md にガイダンスを追加しました。知能とトレードオフになり得る変更については、問題をより早く捕捉できるよう、ソーク期間、より広範な eval スイート、段階的ロールアウトを追加していきます。

最近、プロダクト上の意思決定とその背景にある理由を深く説明する余地を持てるよう、X 上に @ClaudeDevs を開設しました。同じ情報は GitHub 上の集中スレッドでも共有していきます。

最後に、ユーザーの皆さんに感謝したいと思います。`/feedback` コマンドで問題を共有してくださった方々、あるいは具体的で再現可能な例をオンラインに投稿してくださった方々こそが、最終的にこれらの問題を特定し修正することを可能にしてくれました。本日、すべての購読者の利用上限をリセットします。

フィードバックとご辛抱に、心より感謝申し上げます。