16

AI に耐える技術評価を設計する

— Designing AI-resistant technical evaluations —

公開日	2026-01-21
原題	Designing AI-resistant technical evaluations
著者	Anthropic Engineering Team
原文	https://www.anthropic.com/engineering/AI-resistant-technical-evaluations
翻訳	Claude（機械翻訳／Anthropic）
編集	2026-04-09

AI に耐える技術評価を設計する

執筆: Tristan Hume、Anthropic の性能最適化チームのリード。Tristan は、Anthropic で数十人の性能エンジニアを採用する助けとなったテイクホームテストを設計し、そして再設計してきた。

AI の能力が進歩するにつれて、技術候補者を評価することは難しくなります。今日は人間のスキルレベルをうまく区別できるテイクホームが、明日にはモデルに容易に解かれ、評価には使えなくなるかもしれません。

2024 年初頭から、私たちの性能エンジニアリングチームは、候補者が仮想アクセラレータ用のコードを最適化するテイクホームテストを使ってきました。1,000 人超の候補者がこれを終え、そのうち数十人が今ここで働いています——Trainium クラスタを立ち上げ、Claude 3 Opus 以降の全モデルを出荷したエンジニアを含めて。

しかし新しい Claude モデルが出るたびに、このテストを再設計せざるを得なくなりました。同じ制限時間で与えると、Claude Opus 4 はほとんどの人間応募者を上回りました。それでも最強の候補者は区別できていました——しかし次に Claude Opus 4.5 がそれにも並んでしまいました。無制限の時間があれば人間はまだモデルを上回れますが、テイクホームの制約の下では、私たちのトップ候補者と最も有能なモデルの出力を区別する術がなくなってしまったのです。

テイクホームがまだシグナルを運ぶものであり続けるよう、私は今やテイクホームを 3 バージョン反復してきました。そのたびに、どんな評価が AI 支援に対して堅牢で、どんなものがそうでないのか、新しいことを学びました。

本稿では、オリジナルのテイクホーム設計、各 Claude モデルがどうそれを打ち破ったか、そしてテストをトップモデルの能力より前にあり続けさせるために取らざるを得なかったますます変わったアプローチを記述します。私たちの仕事はモデルと共に進化してきましたが、まだ多くの強いエンジニアが必要です——見つけ方がますます創造的になるだけで。

その目的のため、オリジナルのテイクホームをオープンなチャレンジとして公開します。無制限の時間があれば、人間の最高性能はまだ Claude が達成できるものを超えるからです。Opus 4.5 を打ち負かせた方は、ぜひ連絡をください——詳細は本稿の末尾にあります。

テイクホームの起源

2023 年 11 月、Claude Opus 3 の訓練とローンチを準備していました。新しい TPU と GPU クラスタを確保し、大型の Trainium クラスタが到着予定で、アクセラレータに過去と比べて大幅に多くを費やしていたものの、新しいスケールに対して性能エンジニアが不足していました。私は Twitter で投稿 し、メールで連絡を求

めました。それは、標準の採用パイプライン（スタッフと候補者の時間を大きく消費するプロセス）で評価しきれないほど有望な候補者をもたらしました。

候補者をより効率的に評価する方法が必要でした。そこで、役割の要求を適切に捉え、最も能力の高い応募者を特定できるテイクホームテストを設計するのに 2 週間を費やしました。

設計目標

テイクホームには悪い評判があります。たいていは汎用的な問題で満ちており、エンジニアが退屈に感じ、フィルターとしても働きません。私の目標は違いました——候補者がやる気を出して参加し、技術スキルを高い解像度で捉えられる、本当に魅力的なものを作ることでした。

この形式は、性能エンジニアリングスキルの評価に関してライブ面接に対して利点も提供します。

長い時間軸: エンジニアは 1 時間未満の締め切りに直面することは稀です。4 時間枠（後に 2 時間に短縮）は、実際の仕事の性質をよりよく反映します。ほとんどの実タスクよりはまだ短いですが、負担の大きさとバランスを取る必要があります。

現実的な環境: 誰かが見ていて実況を期待されることもありません。候補者は邪魔なく自分のエディタで作業します。

理解とツーリングのための時間: 性能最適化は既存システムの理解と、時にはデバッグツールの構築を必要とします。どちらも通常の 50 分面接では現実的に評価しづらいです。

AI 支援との互換性: Anthropic の 一般候補者ガイダンス は、明示されない限り AI なしでテイクホームを完了するよう求めます。このテイクホームでは、明示的に別だと伝えています。

長い時間軸の問題は AI が完全に解くのが難しいので、候補者は（仕事でそうするように）AI ツールを使いつつも、自分自身のスキルを示す必要があります。

こうした形式固有の目標を超えて、面接を設計するときの原則をテイクホームにも適用しました。

実務を代表する: 問題は候補者に仕事が実際に何を含むかの味見を与えるべきです。

高シグナル: 単一の洞察に依存する問題は避け、候補者に能力を十分見せる機会を多数与え、偶然に任せる余地を最小化すべきです。スコア分布が広く、強い候補者でもすべてを終えられないほどの深さが必要です。

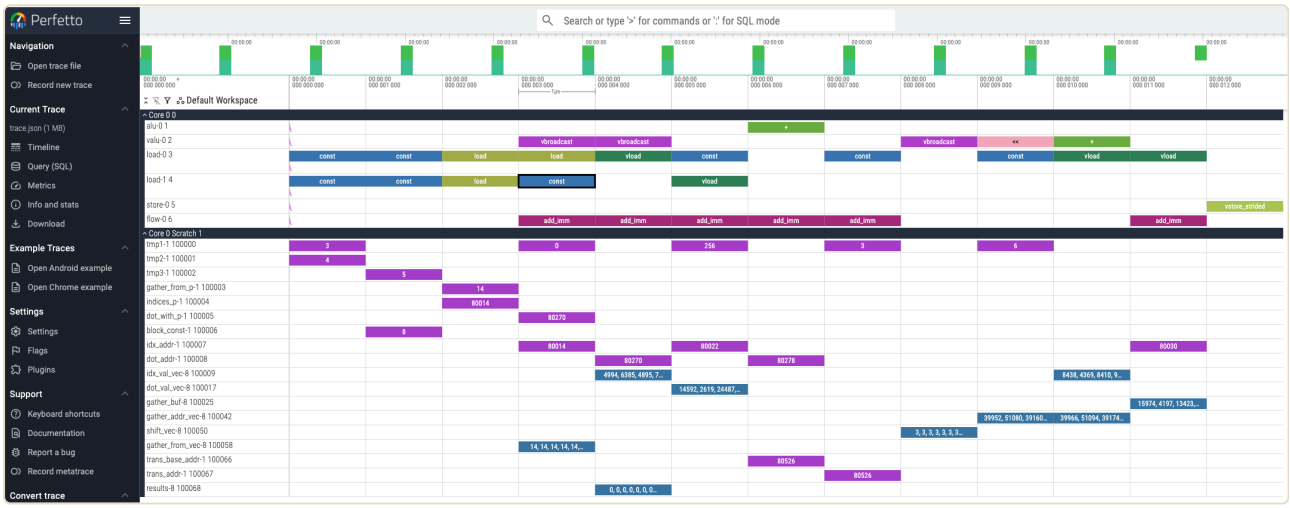
特定のドメイン知識なし: 良いファンダメンタルズを持つ人は仕事で詳細を学べます。狭い専門性を要求すれば候補者プールを不必要に狭めます。

楽しい: 速い開発ループ、深みのある興味深い問題、創造性の余地。

仮想マシン

私は TPU に似た特性を持つ架空のアクセラレータ用の Python シミュレータを作りました。候補者はホットリロードの Perfetto トレース（Trainium 上のツーリング に似た、すべての命令を示すもの）を使って、このマシン上で走るコードを最適化します。

このマシンには、アクセラレータ最適化を興味深くする機能を含めました——手動管理のスクラッチパッドメモリ（CPU と違い、アクセラレータはしばしば明示的なメモリ管理を要する）、VLIW（各サイクルで並列に動作する複数の実行ユニット、効率的な命令パッキングを要する）、SIMD（命令あたり多数の要素へのベクトル操作）、マルチコア（コア間で作業を分散）。



タスクは並列木走査で、ほとんどの性能エンジニアはまだディープラーニングで働いたことがなく仕事でドメイン詳細を学べるため、意図的にディープラーニング寄りにはしませんでした。問題は、古典的な ML 最適化課題へのオマージュとしてのブランチレス SIMD 決定木推論に触発されたもので、以前出会ったことがある候補者はわずかでした。

候補者は完全に逐次な実装から始め、徐々にマシンの並列性を活用していきます。ウォームアップはマルチコア並列化、その後候補者は SIMD ベクトル化か VLIW 命令パッキングのどちらに取り組むかを選びます。オリジナル版には、候補者がまずデバッグする必要があるバグも含まれており、ツーリングを組み立てる能力を試せました。

初期の成果

初期のテイクホームはうまく機能しました。Twitter バッチのある 1 人は、他の全員より実質的に高くスコアしました。彼は 2 月初旬、標準パイプライン経由の最初の採用者の 2 週間後に始めました。テストは予測的でした——彼はすぐにカーネル最適化を始め、テンソルインデックスの数学が 32 ビットをオーバーフローする、ローンチをブロックしていたコンパイラバグの回避策を見つけました。

それから 1 年半で、約 1,000 人の候補者がテイクホームを終え、現在の性能エンジニアリングチームの大半の採用に役立ちました。書類では経験が限られている候補者に特に価値がありました——最も高性能なエンジニアの何人かは大学卒業直後から来ましたが、自信を持って採用できるほどテイクホームでスキルを示しました。

フィードバックは肯定的でした。多くの候補者は楽しんでいたために 4 時間制限を超えて作業しました。無制限時間の最強の提出には、完全な最適化ミニコンパイラや、私が予想しなかったいくつかの賢い最適化が含まれていました。

そして Claude Opus 4 がそれを打ち破った

2025 年 5 月までに、Claude 3.7 Sonnet は既に 50% 超の候補者が Claude Code に完全に委譲した方がよくなるレベルまで忍び寄っていました。その後、私はプレリリース版の Claude Opus 4 をテイクホームでテストしました。4 時間制限内でほぼすべての人間より最適化された解を思いつきました。

これは Claude モデルに敗れた最初の面接ではありません。2023 年、当時の Claude モデルが豊富に知識を持つ一般的タスクに基づいた質問では簡単に解けてしまったので、特にライブ面接質問を設計しました。知識よりも問題解決スキルを要求する質問を、実在する（しかしニッチな）仕事で解いた問題に基づいて設計しようとしました。Claude 3 Opus はその質問のパート 1 を打ち破り、Claude 3.5 Sonnet はパート 2 を打ち破りました。他のライブ質問も AI 耐性がないので、それでもまだ使っています。

テイクホームには直截的な修正がありました。問題は誰も 4 時間では探索しきれないほどの深みがあったので、Claude Opus 4 を使って苦戦し始める箇所を特定しました。そこがバージョン 2 の新しい開始点になりました。よりクリーンなスターターコードを書き、より深みのための新しいマシン機能を追加し、マルチコアを取り除きました（Claude がすでに解いていて、シグナルを足さず開発ループを遅くするだけだったので）。

また、制限時間を 4 時間から 2 時間に短縮しました。もともと 4 時間を選んだのは、バグや混乱で少し行き詰まった場合に沈没するリスクを減らすことを好む候補者のフィードバックに基づいたものでしたが、スケジュール調整のオーバーヘッドがパイプラインに数週間の遅延を招いていました。2 時間の方が週末に収めやすいのです。

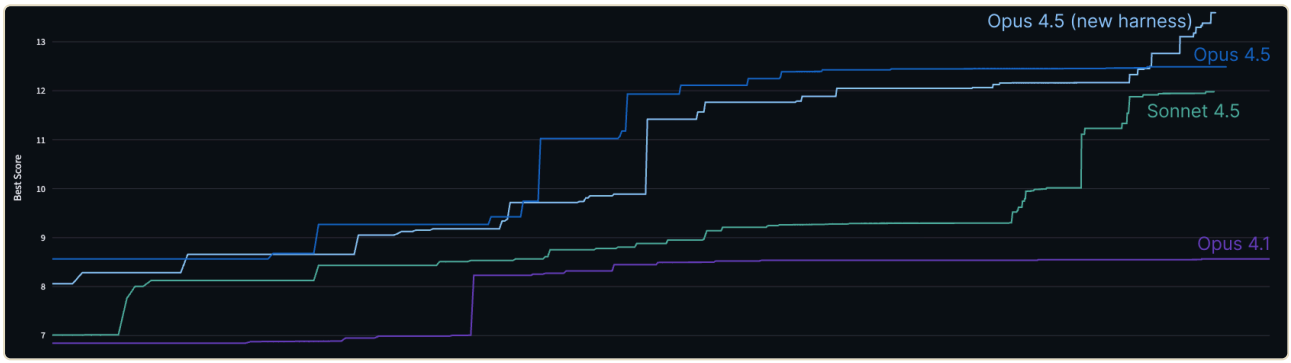
バージョン 2 はデバッグやコード量よりも、賢い最適化の洞察を強調しました。これは数ヶ月間役立ちました。

そして Claude Opus 4.5 がそれを打ち破った

プレリリース版 Claude Opus 4.5 のチェックポイントをテストしたとき、Claude Code が 2 時間問題に取り組み、徐々に解を改善していくのを見ました。初期のボトルネックを解き、よくある微最適化をすべて実装し、1 時間以内に合格しきい値に達しました。

その後、打ち破れないメモリ帯域幅ボトルネックに当たったと確信して止まりました。ほとんどの人間も同じ結論に達します。しかしそのボトルネックを回避するために問題構造を活用する賢い技があります。Claude に達成可能なサイクル数を伝えると、しばらく考えてその技を見つけました。その後デバッグし、チューニングし、さらなる最適化を実装しました。2 時間の時点で、そのスコアは制限時間内の最良の人間性能——しかもそれは Claude 4 を強力に操縦した人間のもの——に並びました。

より厳密に内部のテスト時計算ハーネスで試し、2 時間で人間に勝ち、時間と共に登り続けられることを確認しました。ローンチ後にハーネスを汎用的に改善し、さらに高いスコアを得ました。



問題が発生していました。私たちは、テイクホームで最良の戦略が Claude Code への委譲になるモデルをリリースしようとしていたのです。

選択肢を考える

同僚の中には AI 支援の禁止を提案する者もいました。私はそうしたくありませんでした。施行上の難しさを超えて、人々が仕事で依然として不可欠な役割を果たし続けている以上、仕事でそうするように AI のある *環境* で候補者が自分を際立たせる方法を *何らかの* 形で考え出せるはずだ、という感覚がありました。数時間より長いタスクでしか人間に優位がない、という考え にまだ屈したくなかったのです。

他の人は「Claude Code が単独で達成するものを実質的に上回る」水準まで基準を引き上げることを提案しました。ここでの懸念は、Claude が速く働くことです。人間は典型的に、最適化を始める前に 2 時間の半分を問題の読解と理解に費やします。Claude を操縦しようとする人間は常に後追いになり、Claude がしたことを事後的にしか理解できない可能性が高いです。支配的な戦略が、座って眺めることになりかねません。

今日、Anthropic の性能エンジニアにはまだたくさんの仕事がありますが、それは厳しいデバッグ、システム設計、性能分析、システムの正確性の検証方法、Claude のコードをよりシンプルで上品にする方法のようなものに見えます。残念ながら、これらのことは大量の時間や共通文脈なしに客観的にテストするのが難しいです。面接を仕事の代表物として設計するのは常に難しかったのですが、今はかつてないほど難しいのです。

しかし、もし新しいテイクホーム設計に投資したら、それも Claude Opus 4.5 が解いてしまうか、あるいは人間が 2 時間では完了不可能なほど難しくなってしまうのではないかと心配でした。

試行 1: 異なる最適化問題

Claude が設計したものを素早く実装するのを助けてくれるので、より難しいテイクホームを開発しようという気になりました。Anthropic で自分が行ったより難しいカーネル最適化の 1 つ——バンク衝突を避けつつ 2D TPU レジスタ上で効率的にデータを 転置 する——に基づく問題を選びました。それを仮想マシン上のより単純な問題に蒸留し、Claude に 1 日未満で変更を実装させました。

Claude Opus 4.5 は私が考えもしなかった素晴らしい最適化を見つけました。注意深い分析を通じて、データを転置する方法を考える代わりに計算全体を転置できると気づき、それに従ってプログラム全体を書き直したのです。

実際のケースではこれはうまくいかなかったので、問題にパッチを当ててそのアプローチを取り除きました。Claude はその後進展しましたが、最も効率的な解は見つけられませんでした。新しい問題が見つかったようで、あとは人間候補者が十分速くそれに到達することを祈るだけ、と思いました。しかし何かもやもやした疑念があったので、Claude Code の「ultrathink」機能をより長い思考予算で使って再確認したところ……解けたのです。バンク衝突の修正の技さえ知っていました。

振り返ると、これは試すべき正しい問題ではありませんでした。多くのプラットフォームのエンジニアがデータ転置とバンク衝突で苦戦してきており、Claude はそこから引き出せる訓練データを豊富に持っているのです。私は第一原理から解を見つけましたが、Claude はより大きな経験の道具箱を引き出せたのです。

試行 2: もっと変にする

人間の推論が Claude のより大きな経験ベースに勝てる問題が必要でした——分布外に十分離れた何かです。残念ながら、これは仕事に似たものであるという目標と衝突しました。

私は楽しんだ最も変わった最適化問題について考え、Zachtronics のゲーム に行き着きました。これらのプログラミングパズルゲームは、珍しく強く制約された命令セットを使い、非慣習的な方法でプログラムさせます。たとえば Shenzhen I/O では、プログラムは複数の通信するチップにまたがって分割され、各チップは約 10 命令と 1〜2 のステートレジスタしか持ちません。賢い最適化にはしばしば、命令ポインタや分岐フラグにステートをエンコードすることが含まれます。

小さく強く制約された命令セットを使い、最小命令数のために解を最適化するパズルから成る新しいテイクホームを設計しました。中程度の難易度のパズル 1 つを実装し、Claude Opus 4.5 でテストしました。失敗しました。さらにパズルを埋め、私ほど問題に浸っていない同僚に Claude より上回れるか検証してもらいました。

Zachtronics のゲームとは異なり、意図的に可視化やデバッグツールを提供しませんでした。スターターコードは解が有効かどうかだけをチェックします。デバッグツールを構築することはテストされる一部です——注意深く作った print 文を挿入することも、コーディングモデルに数分でインタラクティブなデバッガを生成させることもできます。ツーリングにどう投資するかの判断も、シグナルの一部です。

私は新しいテイクホームにそこそこ満足しています。より独立したサブ問題から成るため、オリジナルより分散が低いかもしれません。初期結果は有望です——スコアは候補者の過去の仕事の力量とよく相関し、最も有能な同僚の 1 人が今のところどの候補者よりも高くスコアしました。

オリジナルのリアリズムと多様な深みを諦めたのは今でも悲しいです。しかしリアリズムは、もはや私たちが持てない贅沢かもしれません。オリジナルは実務に似ていたからうまく機能しました。代替は新奇な作業をシミュレートするからうまく機能します。

オープンチャレンジ

オリジナルのテイクホームを、無制限の時間で誰でも試せるように公開します。人間専門家は、十分に長い時間軸では現在のモデルに対する優位を保っています。これまで提出された最速の人間解は、広範なテスト時計算をしても Claude が達成したものを大きく上回ります。

公開バージョンはスクラッチから始めますが（バージョン 1 のように）、バージョン 2 の命令セットと単一コア設計を使うので、サイクル数はバージョン 2 と比較可能です。

性能ベンチマーク（仮想マシンのクロックサイクルで計測）：

- **2164 サイクル**: テスト時計算ハーネスで何時間もかけた Claude Opus 4
- **1790 サイクル**: カジュアルな Claude Code セッションでの Claude Opus 4.5、約 2 時間で最良の人間性能に並ぶ
- **1579 サイクル**: 私たちのテスト時計算ハーネスで 2 時間かけた Claude Opus 4.5
- **1548 サイクル**: 2 時間を大きく超えるテスト時計算の Claude Sonnet 4.5
- **1487 サイクル**: ハーネスで 11.5 時間の Claude Opus 4.5
- **1363 サイクル**: 改善されたテスト時計算ハーネスで何時間もかけた Claude Opus 4.5

GitHub からダウンロード。1487 サイクル未満に最適化してローンチ時の Claude の最良性能を上回れたら、performance-recruiting@anthropic.com にコードと履歴書を送ってください。

あるいは 通常のプロセス経由で応募 することもできます——（現在の）Claude 耐性テイクホームを使います。それがどれだけ持つか、私たちも気になるところです。