04

"think" ツール: 複雑なツール利用で Claude に立ち止まって考えさせる

— The "think" tool: Enabling Claude to stop and think in complex tool use situations —

公開日	2025-03-20
原題	The "think" tool: Enabling Claude to stop and think in complex tool use situations
著者	Anthropic Engineering Team
原文	https://www.anthropic.com/engineering/claude-think-tool
翻訳	Claude（機械翻訳／Anthropic）
編集	2026-04-09

"think" ツール: 複雑なツール利用で Claude に立ち止まって考えさせる

- 拡張思考（Extended thinking）に関する更新

2025 年 12 月 15 日

拡張思考の能力は最初のリリース以降改善されてきたため、現在では多くのケースで専用の think ツールではなく拡張思考機能の使用を推奨しています。拡張思考は、Claude に複雑な問題を推論する余地を与えるという同様の利点を、より良い統合性と性能で提供します。実装の詳細は拡張思考のドキュメントを参照してください。

Claude の複雑な問題解決能力を継続的に高めていくなかで、私たちは特に効果的な手法を発見しました。複雑なタスク中に構造化された思考のための専用スペースを作る「think」ツールです。

このシンプルかつ強力な手法は（後述するように Claude の新機能「拡張思考（extended thinking）」とは別物です。実装詳細はこちら）、Claude のエージェント型ツール利用能力を大幅に改善しました。ポリシーの遵守、一貫した意思決定、多段階問題の処理といったことが、最小限の実装負荷で可能になります。

本稿では、検証済みのベンチマーク結果に基づいて、さまざまなアプリケーションで「think」ツールを実装する方法と、開発者向けの実用的なガイダンスを紹介します。

「think」ツールとは何か

「think」ツールは、最終的な回答に至るプロセスの一部として、専用のスペースを持った追加の思考ステップを Claude に付与する仕組みです。

名前は似ていますが、拡張思考とは別の概念です。拡張思考は Claude が応答を生成し始める *前* に行うことに関するものです。拡張思考により Claude は行動を起こす前に計画を深く考え反復します。「think」ツールは、Claude が応答を生成し始めた *後* に、次へ進むために必要な情報がすべて揃っているかを立ち止まって考えるためのステップを追加するものです。これは、長いツール呼び出しの連鎖を実行している時や、ユーザーとの長い多段階会話の中で特に有用です。

そのため「think」ツールは、ユーザーのクエリだけでは応答を組み立てるのに必要な情報が揃わず、外部情報（たとえばツール呼び出しの結果）を処理する必要があるケースに向いています。Claude が「think」ツールで行う推論は、拡張思考で得られるものほど包括的ではなく、むしろモデルが発見する *新しい* 情報に焦点を当てたものになります。

非逐次的なツール呼び出しや単純な指示追従のような、より単純なツール利用シナリオには拡張思考を推奨します。拡張思考は、Claude がツールを呼び出す必要がないコーディング・数学・物理といったユースケースにも有用です。一方「think」ツールは、Claude が複雑なツールを呼び出す必要がある、長いツール呼び出し連鎖でツールの出力を注意深く分析する必要がある、詳細なガイドラインを伴うポリシー重視の環境を進む必要がある、あるいは前のステップに積み上がる形で逐次的な意思決定を行う必要があり（各ミスがコスト大の）場合に適しています。

τ-Bench で提供されている標準的なツール仕様フォーマットを使ったサンプル実装を示します。

```
{
  "name": "think",
  "description": "Use the tool to think about something. It will not obtain new info
rmation or change the database, but just append the thought to the log. Use it when
complex reasoning or some cache memory is needed.",
  "input_schema": {
    "type": "object",
    "properties": {
      "thought": {
        "type": "string",
        "description": "A thought to think about."
      }
    },
    "required": ["thought"]
  }
}
```

τ-Bench での性能

私たちは「think」ツールを **τ-bench（tau-bench）** で評価しました。τ-bench は、リアルなカスタマーサービスシナリオでモデルがツールを使う能力を測るための包括的なベンチマークで、「think」ツールはその評価環境の一部として組み込まれています。

τ-bench は Claude の以下の能力を測ります。

- シミュレートされたユーザーとの現実的な会話を進める
- 複雑なカスタマーサービス担当者向けポリシー・ガイドラインに一貫して従う
- 環境データベースへのアクセス・操作のためにさまざまなツールを使う

τ-bench の主要な評価指標は **pass^k** で、1 つのタスクに対する k 回の独立試行すべてが成功する確率を、全タスクで平均して測ります。多くの LLM 評価で一般的な pass@k 指標（k 回の試行のうち少なくとも 1 回成功するか）とは異なり、pass^k は一貫性と信頼性を評価します。ポリシーへの一貫した遵守が不可欠なカスタマーサービス用途にとって、これは決定的に重要な性質です。

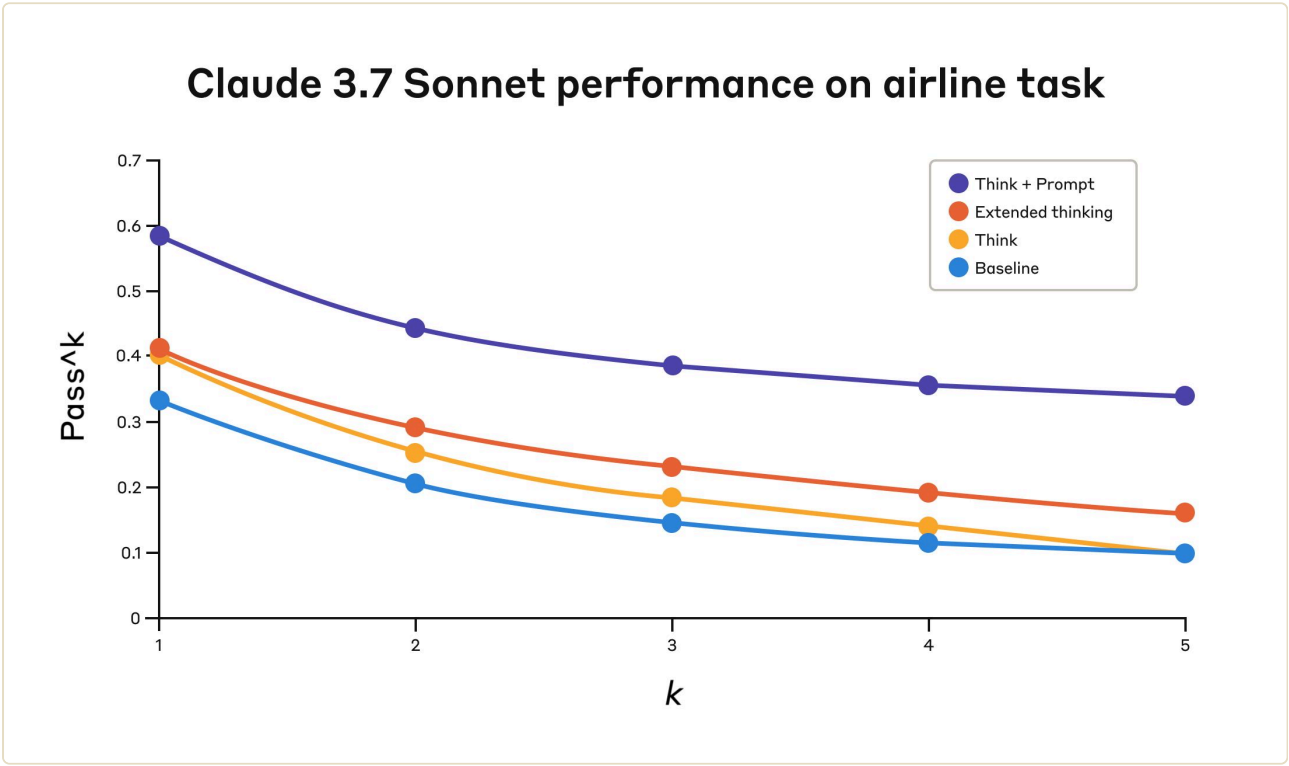
性能分析

私たちは以下の構成を比較評価しました。

1. ベースライン（「think」ツールも拡張思考モードも無し）
2. 拡張思考モードのみ
3. 「think」ツールのみ
4. 「think」ツール + 最適化されたプロンプト（航空会社ドメイン用）

その結果、ベンチマークの「航空会社（airline）」ドメインと「小売（retail）」ドメインの両方で、Claude 3.7 が「think」ツールを効果的に使ったときに劇的な改善が見られました。

- **航空会社ドメイン**: 最適化プロンプト付きの「think」ツールは pass^1 指標で 0.570 を達成し、ベースラインの 0.370 に対して相対 54% の改善。
- **小売ドメイン**: 「think」ツール単体で 0.812 を達成し、ベースラインは 0.783。



Tau-Bench の「航空会社」ドメインにおける Claude 3.7 Sonnet の 4 構成での性能。

Claude 3.7 Sonnet の Tau-Bench 航空会社ドメインでの性能

構成	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
「Think」+ プロンプト	0.584	0.444	0.384	0.356	0.340
「Think」	0.404	0.254	0.186	0.140	0.100
拡張思考	0.412	0.290	0.232	0.192	0.160
ベースライン	0.332	0.206	0.148	0.116	0.100

4 つの構成による評価結果。スコアは比率。

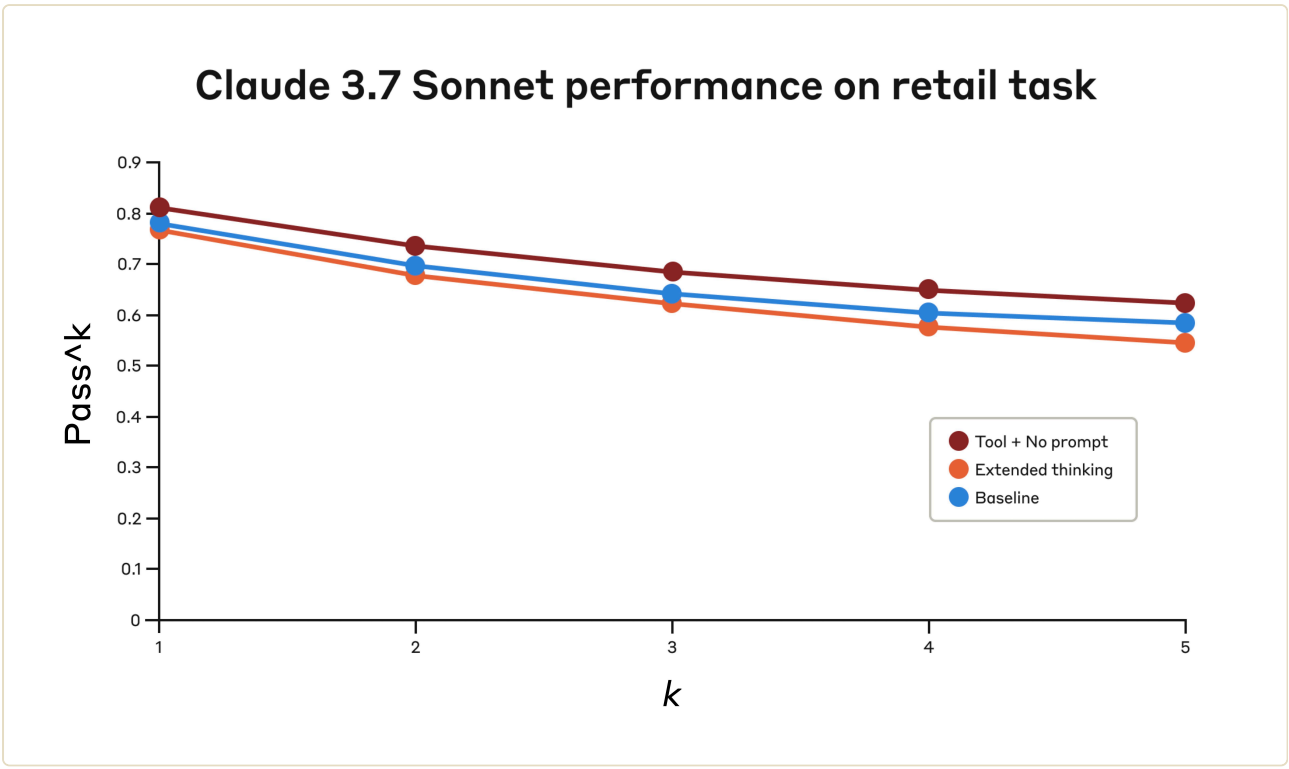
航空会社ドメインで最高性能を叩き出したのは、「think」ツールと、顧客要求を分析するときに使う推論アプローチの例を含んだ最適化プロンプトを組み合わせた構成でした。以下に最適化プロンプトの例を示します。

```
## Using the think tool

Before taking any action or responding to the user after receiving tool results, use
the think tool as a scratchpad to:
- List the specific rules that apply to the current request
- Check if all required information is collected
- Verify that the planned action complies with all policies
- Iterate over tool results for correctness

Here are some examples of what to iterate over inside the think tool:
<think_tool_example_1>
User wants to cancel flight ABC123
- Need to verify: user ID, reservation ID, reason
- Check cancellation rules:
  * Is it within 24h of booking?
  * If not, check ticket class and insurance
- Verify no segments flown or are in the past
- Plan: collect missing info, verify rules, get confirmation
</think_tool_example_1>

<think_tool_example_2>
User wants to book 3 tickets to NYC with 2 checked bags each
- Need user ID to check:
  * Membership tier for baggage allowance
  * Which payments methods exist in profile
- Baggage calculation:
  * Economy class × 3 passengers
  * If regular member: 1 free bag each → 3 extra bags = $150
  * If silver member: 2 free bags each → 0 extra bags = $0
  * If gold member: 3 free bags each → 0 extra bags = $0
- Payment rules to verify:
  * Max 1 travel certificate, 1 credit card, 3 gift cards
  * All payment methods must be in profile
  * Travel certificate remainder goes to waste
- Plan:
1. Get user ID
2. Verify membership level for bag fees
3. Check which payment methods in profile and if their combination is allowed
4. Calculate total: ticket price + any bag fees
5. Get explicit confirmation for booking
</think_tool_example_2>
```

特に興味深いのは、アプローチ間の比較です。最適化プロンプトと「think」ツールの組み合わせは、拡張思考モード（プロンプト無しの「think」ツールと似た性能）より著しく良い結果を出しました。「think」ツール単体（プロンプト無し）ではベースラインより改善しましたが、最適化アプローチには及びませんでした。

「think」ツール + 最適化プロンプトの組み合わせが大差で最強だったのは、ベンチマークの 航空会社ポリシー 部分が非常に複雑で、モデルに「どう考えるか」の例を与えることが最も効いたからだと考えられます。

小売ドメインでも、各アプローチの影響を理解するために複数の構成をテストしました。



Tau-Bench の「小売」ドメインにおける Claude 3.7 Sonnet の 3 構成での性能。

Claude 3.7 Sonnet の Tau-Bench 小売ドメインでの性能

構成	*k*=1	*k*=2	*k*=3	*k*=4	*k*=5
「Think」+ プロンプト無し	0.812	0.735	0.685	0.650	0.626
拡張思考	0.770	0.681	0.623	0.581	0.548
ベースライン	0.783	0.695	0.643	0.607	0.583

3 つの構成による評価結果。スコアは比率。

「think」ツールは追加プロンプト無しでも pass^1 の最高スコア 0.812 を達成しました。小売ポリシー は航空会社ドメインと比べると明らかに navigation しやすく、Claude はさらなる指示なしに「考えるスペース」を得るだけで改善できたのです。

τ-Bench 分析からの主要な知見

私たちの詳細な分析から、「think」ツールを効果的に実装するのに役立ついくつかのパターンが見えてきました。

1. **困難なドメインではプロンプトが大きく影響する。**「think」ツールをただ使えるようにするだけでもある程度改善しますが、最適化プロンプトと組み合わせれば、困難なドメインでは劇的に良い結果が得られます。一方、易しいドメインでは単に「think」へのアクセスがあるだけで恩恵が得られることがあります。
2. **試行間の一貫性の向上。**「think」を使うことによる改善は、k=5 までの pass^k で維持されました。これは、このツールが Claude のエッジケースや異常シナリオの処理をより効果的に助けていることを示しています。

SWE-Bench での性能

同様の「think」ツールは、Claude 3.7 Sonnet を評価する際の SWE-bench セットアップにも追加され、最先端スコア 0.623 の達成に貢献しました。適応版の「think」ツール定義を以下に示します。

```
{
  "name": "think",
  "description": "Use the tool to think about something. It will not obtain new info
rmation or make any changes to the repository, but just log the thought. Use it when
complex reasoning or brainstorming is needed. For example, if you explore the repo a
nd discover the source of a bug, call this tool to brainstorm several unique ways of
fixing the bug, and assess which change(s) are likely to be simplest and most effect
ive. Alternatively, if you receive some test results, call this tool to brainstorm w
ays to fix the failing tests.",
  "input_schema": {
    "type": "object",
    "properties": {
      "thought": {
        "type": "string",
        "description": "Your thoughts."
      }
    },
    "required": ["thought"]
  }
}
```

私たちの実験（「think」ツール有り n=30 サンプル、無し n=144 サンプル）では、このツールを含めることだけで平均 1.6% の性能改善が見られました（Welch の t 検定: $t(38.89) = 6.71$, $p < .001$, $d = 1.47$）。

「think」ツールを使うべき場面

これらの評価結果から、Claude が「think」ツールから最も恩恵を受ける具体的シナリオを特定しました。

1. **ツール出力の分析。**前のツール呼び出しの出力を注意深く処理してから行動する必要があり、アプローチを後戻りさせる必要があるかもしれない場合。
2. **ポリシー重視の環境。**Claude が詳細なガイドラインに従い、遵守を検証する必要がある場合。

3. **逐次的な意思決定。**各アクションが前のアクションの上に積み上がり、ミスがコスト大の場合（多段階ドメインで頻出）。

実装のベストプラクティス

Claude で「think」ツールを最大限活かすために、τ-bench 実験に基づく以下の実装プラクティスを推奨します。

1. ドメイン特化例による戦略的プロンプト

最も効果的なのは、「think」ツールをいつ・どう使うかの明確な指示を与えることです。τ-bench 航空会社ドメインで使ったようなものです。あなたのユースケースに合わせた例を提供することで、モデルが「think」ツールを使う効果性は大幅に向上します。

- 推論プロセスで期待される詳細度
- 複雑な指示を実行可能なステップに分解する方法
- 一般的なシナリオを処理するための意思決定ツリー
- 必要な情報がすべて揃ったかの確認方法

2. 複雑なガイダンスはシステムプロンプトに配置する

指示が長い／複雑な場合、「think」ツールに関する指示をツール説明そのものに入れるより、システムプロンプトに含めた方が効果的だと分かりました。このアプローチはより広い文脈を与え、モデルが全体の振る舞いに思考プロセスをより良く統合できるようにします。

「think」ツールを使う べきでない 場面

「think」ツールは大きな改善をもたらし得ますが、すべてのツール利用ケースに適用できるわけではなく、プロンプトの長さと出力トークンの増加というコストも伴います。特に、以下のケースでは改善が得られないことが分かっています。

1. **非逐次的なツール呼び出し。**タスク完了のために Claude が単一のツール呼び出しや複数の並列呼び出しだけで済むなら、「think」を入れても改善は見込めません。
2. **単純な指示追従。**Claude が従うべき制約が多くなく、デフォルトの振る舞いで十分な場合、追加の「think」で得られる利得はありません。

始め方

「think」ツールは Claude の実装にわずか数ステップで追加でき、有意義な改善をもたらす素直な追加機能です。

1. **エージェント型ツール利用シナリオでテストする。**Claude がポリシー遵守や長いツール呼び出し連鎖での複雑な推論に苦戦している、難しいユースケースから始めましょう。
2. **ツール定義を追加する。**自分のドメインにカスタマイズされた「think」ツールを実装します。最小限のコードで構造化された推論を可能にします。いつ・どう使うかの指示と、自分のドメインに関連する例をシステムプロンプトに含めることも検討してください。
3. **監視と調整。**Claude が実運用でどうツールを使うかを見て、より効果的な思考パターンを促すようにプロンプトを調整します。

最大の利点は、このツールの追加が性能面でほとんどダウンサイドを持たないことです。Claude が使うと決めない限り外部の振る舞いを変えませんし、既存のツールやワークフローにも干渉しません。

結論

私たちの研究は、複雑なタスクでポリシー遵守と長いツール呼び出し連鎖における推論を要する場合に、「think」ツールが Claude 3.7 Sonnet の性能を大幅に高めることを示しました。「Think」は万能解ではありませんが、適切なユースケースに対しては、最小限の実装複雑性で実質的な恩恵を提供します。

あなたが「think」ツールで、より有能で信頼でき透明性の高い AI システムをどう作るのか楽しみにしています。

---

1. τ-Bench の結果は Claude 3.7 Sonnet と「think」ツールによる改善にフォーカスしましたが、私たちの実験では Claude 3.5 Sonnet (New) も 3.7 Sonnet と同じ構成で性能向上を達成できており、この改善が他の Claude モデルにも一般化することを示しています。