観測モデル — 実際の動作と比較検証の根拠をまとめる

2026 年 11 月 15 日
SimpleModeling

SimpleModeling では、Experiment を骨格に、内部動作の解析に必要な詳細を加えたものを観測モデルと呼びます。入力・実行条件・Arm・Run・結果・評価に、呼び出し構造や I/O 回数、時間・資源消費を結び付け、不具合や性能の調査と、よりよい選択肢の探索に使います。CNCF の自動トレースを活用し、アプリ側の観測用追加実装なく、この共通の根拠を整えます。

1　観測モデルとは

実際の動作を調べ、構成や処理方式を選ぶには、入力と実行条件、使った選択肢、得られた結果を一つの実行について読める形にする必要があります。

観測モデルは、Experiment を骨格として、動作解析に必要な情報を加えたモデルです。Experiment が入力・実行条件・Arm・Run・結果・評価の対応を持ち、そこへ呼び出し構造、I/O 回数、処理区間の時間、CPU 負荷、キャッシュなどの詳細を加えます。

Experiment は、複数の選択肢を実際に動かし、その結果を比較して採用・使い分けを判断するためのモデルです。実利用での比較と Corpus による再実行の両方を扱います。この骨格に内部動作の詳細を重ねることで、不具合や性能の調査と構成比較を、一つの観測モデルでつなげます。

図 1 は、Experiment の骨格に内部動作の詳細を加え、一つの観測モデルから調査と比較へつなぐ構造を示します。

2　観測モデルの利用目的

実行の記録は、起きた問題を調べる根拠にも、変更や候補の効果を比べる根拠にもなります。

観測モデルの利用目的を、次の二つに分けます。

- 不具合の検出と調査
- よりよい構成の探索

不具合の検出と調査では、問題が起きた処理と条件を捉えます。よりよい構成の探索では、候補の品質・性能・総費用を比較します。どちらも、同じ実行に属する条件・結果・負担を結び付けた記録を使います。

2.1　不具合の検出と調査

開発時には、コンパイルや型による検査、テスト・Executable Specification、レビュー、Lint で品質・性能・費用を確認します。しかし、実際の利用条件で生じる二重実行や過剰な I/O、性能低下、過剰な消費は、それらの確認だけでは捉えきれません。

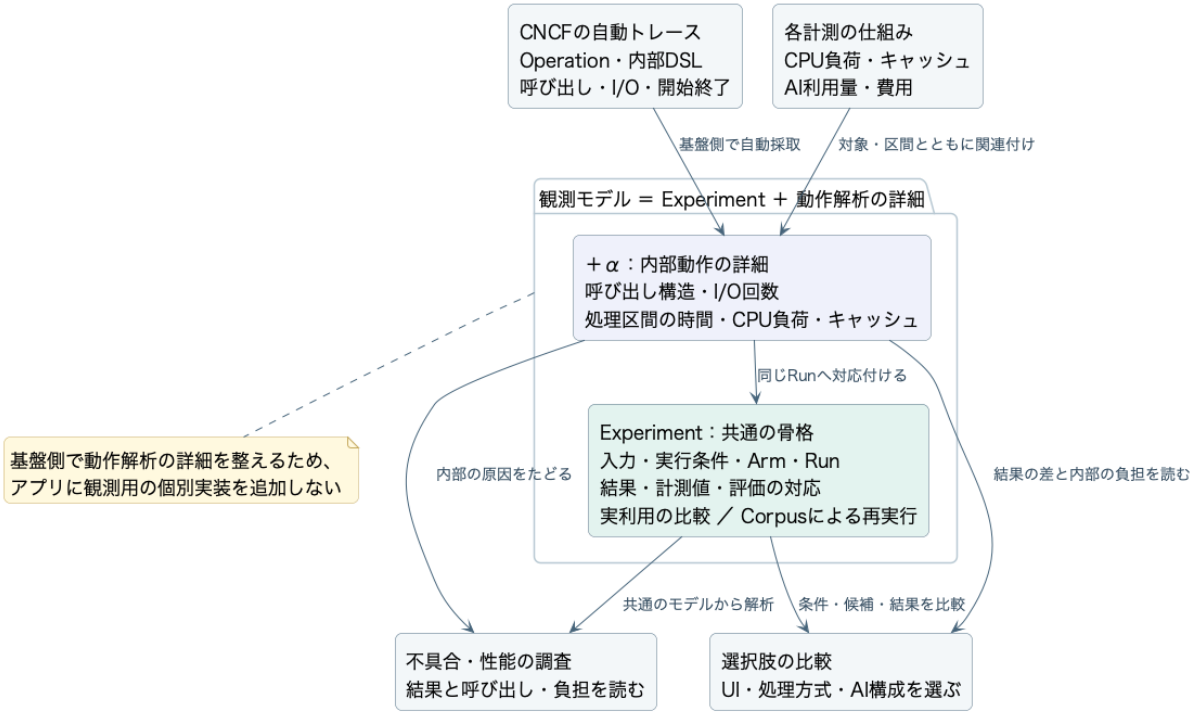

図 1　観測モデルは Experiment の骨格と内部動作の詳細から成る

観測モデルでは、実際の呼び出しと計測値を結び付け、不具合が生じた処理と条件を調べる根拠を用意します。

出力が正しくても、同じ処理を二回実行していれば時間や資源を余分に使います。結果に至る呼び出しの構造・回数と、その区間の時間・消費を合わせて捉えることで、外部から見える結果だけでは分からない動作を調べられます。

AI についても、想定のない呼び出し、想定より長い実行、再試行の増加を対象にします。性能、負荷と無駄な処理、AI の動作と費用の三つの観点から、問題を絞り込みます。

2.2　よりよい構成の探索

画面・操作方法、処理ロジック、AI 構成などの選択肢は、実際に使った結果や処理の負担が異なります。

観測モデルは、Experiment の入力・条件・Arm・Run・結果・評価に内部の動作を対応付け、採用や使い分けを判断する根拠を用意します。

画面の操作完了率やユーザー評価、処理ロジックの正しさや性能、AI の生成品質とコストなど、選択肢に合う結果を比較します。内部の呼び出しや資源消費を合わせて読むことで、結果の差が生じた処理や条件も調べられます。

3　観測モデルを作る意義

トレースやログを分析するには、実行単位をそろえ、条件・呼び出し・結果・計測値を関連付ける準備が必要です。

この準備を共通化し、分析・理解・再利用に使える表現を作ることが、観測モデルを作る意義です。次の三つの価値を持ちます。

- 解析に適した形への整理
- 可視化
- Corpus 化

定型の表現に対して分析方法を再利用し、その同じモデルを図や表で理解します。さらに入力・条件を再実行できるケースへ整えることで、調査と比較を継続できます。

3.1 解析に適した形への整理

トレースやログでは、実行条件、呼び出し、結果、計測値が別々の記録に分かれています。分析のたびにそれらを結び直すと、同じ準備を繰り返すことになります。

解析に適した形を定型化し、記録を整える処理と、そのモデルを分析する処理を分けます。分析方法を、毎回同じ構造の観測モデルへ適用できます。

実行や処理区間の単位で記録を関連付け、性能、呼び出し回数、資源や AI の利用量を同じ表現から調べられるようにします。分析項目を追加するときも、この表現を読む処理として組み立てます。

対象範囲・粒度・計測範囲と元の記録への参照を保持します。これにより、可視化で着目した箇所から詳細な記録をたどり、再実行用ケースの整備にも同じ根拠を使えます。

3.2 可視化

呼び出しや計測値が多い実行では、個々の値に加えて全体の形や偏りを捉えることが、問題を考える手掛かりになります。

観測モデルを図や集計表へ可視化し、動作の全体像と負担の偏りを見渡します。その効果を、次の二つに分けます。

- 人間の直感を生かす
- 議論の文脈を共有する

3.2.1 人間の直感を生かす

重複する呼び出しや、一つの工程に集中する消費は、数値を一件ずつ追うより、構造と分布を合わせて見ると気付きやすくなります。

可視化によって人間の直感を生かし、形や偏りから感じ取った違和感を、問題の発見や改善案の着想につなげます。

図に呼び出し構造・回数・時間と、取得した利用量や費用を示します。着目箇所から元の記録をたどり、違和感の根拠を調べ、変更後の比較へつなげます。

3.2.2 議論の文脈を共有する

対象の処理範囲や条件の認識が異なると、問題の見方や改善案の比較がかみ合いません。

可視化した観測モデルを共有し、対象・範囲・粒度・条件をそろえた議論の文脈を作ります。

図や表にアプリケーション・処理・期間、入力と実行構成、計測範囲を示します。同じ実行の事実を見ながら、どこを変え、何を比べるかを具体的に検討できます。

3.3 Corpus 化

性能改善や AI 構成の効果を繰り返し確かめるには、比較の起点となる入力と条件を再利用できる必要があります。

Corpus は、再実行に使う入力ケースと期待する結果・評価の根拠を、版として固定したものです。同じ版・ケースを使うことで、実装の改修前後や異なる Arm を同じ起点から繰り返し比較できます。

実行の入力やユーザーの判断を評価用のケースとして整え、期待する結果や判断の根拠とともに Corpus の版へまとめます。再実行のたびに新しい Run と実行結果の記録を残し、同じケースに対する動作・品質・性能・費用を比較します。

3.3.1 Corpus でリクエストミックスを定義する

軽い一覧照会と重い帳票生成では、一件あたりの負担が異なります。全体の処理量を比べるには、どの要求をどの割合・規模で流すかをそろえる必要があります。

リクエストミックスは、要求の種類・構成比・各種類の入力規模を定めたものです。これを Corpus に定義し、性能を比較する要求群をそろえます。

ミックスを定義した Corpus の版に、同時実行数・到着頻度・計測期間を対応付けます。その条件で得た全体・種類別のスループットとレイテンシを残し、改善前後や候補を比較します。

4 Experiment の骨格と動作解析の詳細

実行の結果と内部の負担を合わせて調べるため、比較や再実行を支える骨格と、処理の内部を表す詳細を同じ実行へ結び付けます。

観測モデルは、Experiment の骨格と、動作解析のための詳細から構成します。骨格が入力・条件・Arm・Run・結果・評価を対応付け、詳細が呼び出しの構造・回数・時間・資源消費を表します。

Consequence ／ Conclusion で使う Observation は、CNCF 共通のシステム内部の観測情報です。観測モデルは、Experiment の Run に結果・計測値・評価の記録と内部動作の詳細を対応付けるモデル全体を表します。

Run と実行結果の記録を軸に、入力・条件・候補と内部動作を対応付けます。一つの構成での不具合調査も、複数 Arm による比較も同じ形で表し、Corpus による再実行では版・ケースとの対応を加えます。

4.1 Experiment を観測モデルの骨格にする

複数の選択肢を比べるときは、何を動かし、どの条件で、何が得られ、どう評価したかを対応付ける必要があります。

Experiment は、入力・実行条件・Arm・Run・結果・評価を対応付け、複数の選択肢の実行から採用・使い分けを判断するためのモデルです。観測モデルでは、この Experiment を再実行・比較検証の骨格にします。

候補を Arm として定義し、実行時に割り当て、Run ごとに結果・計測値・評価を記録します。その結果を以後の選択へ反映する流れを持ちます。

再実行と比較に用いる基本概念の役割を、次の表にまとめます。

表 1: 再実行・比較に用いる概念

概念	表すもの
Corpus	再実行に使う入力ケースと期待する結果・根拠を、版として固定したもの。
Experiment	入力・実行条件・Arm・Run・結果・評価を対応付け、選択肢を動かして比較し、採用・使い分けを判断するモデル。
Arm	比較する構成や処理方式。
Run	検証の実行。
実行結果の記録	実行に対応する結果・計測値・評価の記録。

Experiment は実際の利用を通じた A/B・マルチアーム検証と、Corpus で入力条件を固定する再実行の両方を扱います。実利用では実行時の入力・割当て条件を残し、Corpus では同じ版・ケースを起点に新しい Run と実行結果の記録を重ねます。

4.2 CNCF が自動採取する実行記録と呼び出し構造

同じ操作が複数回現れたときには、別の要求なのか、一つの要求内の重複や再試行なのかを区別する必要があります。

CNCF は、Operation 評価の実行記録と内部 DSL の呼び出し構造を基盤側で自動採取し、処理の識別と親子関係を捉えます。

component・service・operation、実行・試行・親実行の識別子に、開始・終了・結果・経過時間と CallTree の操作・I/O 階層を結び付けます。アプリ側へ個別の記録処理を追加せずに、どの処理が何を何回呼び、どこで時間を使ったかを調べられます。

4.3 資源・AI 利用量の計測情報を結び付ける

呼び出しの経過時間とプロセス全体の CPU 使用率では、測定範囲が異なります。

資源・AI 利用量の計測情報に、出所と計測範囲、対象の実行との対応を持たせます。

各計測の仕組みから得た CPU 負荷やキャッシュ、AI 利用量・費用へ、対象プロセス・期間・呼び出し・工程を対応付けます。利用量から費用を算定する場合は、単価と算定範囲も残し、何についての値かを明らかにして分析します。

4.4 Arm・ケース・実行を対応付ける

一回の実行と、その途中の再試行や工程を混ぜると、候補ごとの時間や費用を正しく集計しにくくなります。

入力・実行条件・Arm・Run と、その結果・計測値・評価の記録を対応付け、その同じ実行へ動作解析の詳細を結び付けます。

実行と親子関係・処理区間の識別子を使い、呼び出し階層や計測値を Run と実行結果の記録からたどれるようにします。実利用の比較では候補の割当て条件を、Corpus 再実行では版・ケースとリクエストミックスなどの比較条件を加えます。

5　CNCF の自動トレースでアプリ側の追加実装なく各種分析を行う

アプリごとに記録処理を作ると、取得項目や形式をそろえる作業が業務処理の開発に加わります。

CNCF の自動トレースと共通のモデル化・分析を使い、アプリ側へ観測用の個別実装を追加せずに各種分析を行います。

基盤側で Operation 評価の識別子・開始・終了・結果を採取し、内部 DSL から操作・I/O の呼び出し構造を得ます。共通処理で実行と親子関係をそろえ、観測モデルへまとめることで、処理時間、呼び出し回数、重複や I/O 反復を分析できます。

分析項目は、この共通モデルを読む処理として追加します。CPU・キャッシュや AI 利用量・費用は、各計測の仕組みから得た値を対象実行や区間へ関連付けて使います。観測データの整備と各種分析を共通基盤へ集約し、アプリ側の追加実装を省きます。

6　実利用の観測と Corpus による再実行

実際の利用で起きる動作と、入力条件をそろえたときの差の両方を捉えると、問題の調査と選択肢の比較をつなげられます。

Experiment を骨格とする観測モデルを、実利用の記録と Corpus による再実行の二つの方法で整えます。両方を、不具合の調査とよりよい構成の探索に使います。

- 実利用の記録を残して分析する
- Corpus を使った再実行

実利用では入力条件と実行時に割り当てた Arm を記録し、Run に結果・計測値・評価を対応付けます。Corpus では版・ケースを固定して候補を再実行します。どちらも同じ骨格へ呼び出し構造や資源消費の詳細を加え、可視化や分析に使います。

図 2 では、保存した実行を調べる経路と、Corpus から新たに実行して調べる経路を並べています。どちらも、不具合の調査と構成の探索に使います。

6.1　実利用の記録を残して分析する

処理時間や AI 費用は入力・実行条件によって変わります。時間や消費が増えた原因を調べるには、そのときの呼び出しと条件を残しておく必要があります。

実利用の入力・条件・Arm・Run と、その結果・計測値・評価を記録し、起きたことを分析します。A/B・マルチアーム検証では、実行時の候補の割当てと結果の対応を使って比較します。

オンデマンドまたはリアルタイムで得た呼び出し・時間・資源消費・AI 利用量を Run と実行結果の記録へ対応付けます。保存した記録から不具合や偏りを調べ、同じ評価軸で Arm ごとの結果を読

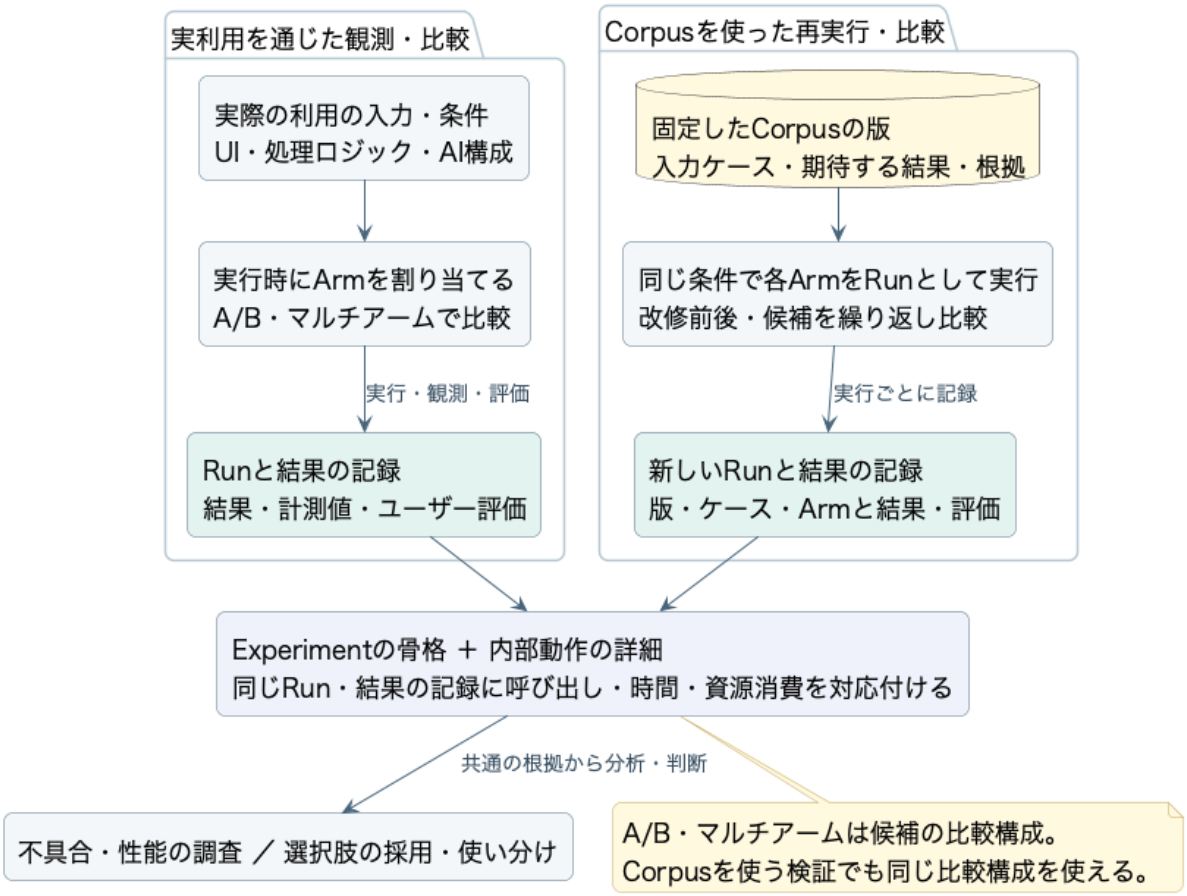

図 2　実利用と Corpus 再実行から同じ骨格へ記録する

み取ります。

6.2　Corpus を使った再実行

構成とともに入力や条件まで変わると、得られた結果に両方の影響が混ざります。

Corpus の版・ケースと比較条件を固定し、選択した Arm を再実行して、新しい Run と、その結果・計測値・評価の記録を得ます。

改修前後の処理や複数の AI 構成を Arm として実行し、ケース・Arm・Run と結果・計測値・評価を対応付けます。性能比較では同じリクエストミックスと実行条件を使い、AI 比較では同じ生成品質の基準とコストの算定範囲を使います。

7　不具合の検出と調査

出力が得られていても、遅延や処理量不足、余分な資源・AI 費用の消費が起きることがあります。

結果に至る過程を調べるため、性能、負荷と無駄な処理、AI の動作と費用という三つの観点を扱います。

以下の観点から、どの動作と負担を捉えるかを整理します。

- 性能：レイテンシとスループット

- 負荷と無駄な処理：CPU・I/O・重複・反復
- AI の動作と費用：呼び出し・実行時間・利用量

7.1 性能：レイテンシとスループット

一件ずつは速くても、要求が集中すると処理が滞る場合があります。要求の種類や入力規模によっても、全体の処理量は変わります。

性能は、レイテンシとスループットの両方を、リクエストミックスと実行条件に対応付けて捉えます。

要求ごとの時間に加え、一定期間の全体・種類別の完了件数を残します。Corpus の版・ミックス、同時実行数・到着頻度・計測期間をそろえた実績から、遅い要求や処理量が落ちる条件を調べます。

7.2 負荷と無駄な処理：CPU・I/O・重複・反復

二重実行や一括化できる I/O の反復は、出力が同じでも時間と資源を使います。処理が進まないまま消費だけが増える場合もあります。

呼び出し構造・回数と資源消費・進捗を合わせ、重複・過剰な反復・暴走の発生箇所を調べます。

CPU 負荷、I/O 回数、キャッシュのヒット率、再試行と消費の増え方を、対象の実行や処理区間へ対応付けます。どの呼び出しが負担を生み、進捗と消費がどう関係しているかを調べる根拠にします。

7.3 AI の動作と費用：呼び出し・実行時間・利用量

AI を呼ばない想定の処理で呼ばれる、想定より長い時間動く、再試行が増えるといった現象は、最終結果だけでは分かりにくい場合があります。

AI の呼び出し条件・構成・時間・利用量を結果へ結び付け、動作と費用の問題を調べます。

呼び出し元・入力条件・回数・時間を残し、提供会社、モデルと版、リーズニングレベル、工程配置へ対応付けます。長い入力や再試行による影響を、試行別・工程別の利用量・費用と累積から読み取ります。

意図しない AI 呼び出しや長時間の実行を調べるため、実行条件と時間・費用を結び付けます。候補比較では同じ算定範囲の実績を用意し、動作の問題と構成の違いによる負担を分けて読み取ります。

8 よりよい構成の探索

画面の操作方法や処理ロジック、AI 構成の選択には、実際に動かした結果を共通の基準で比べる根拠が必要です。

Experiment で選択肢を Arm として定め、実行時に割り当て、結果を観測・評価して、採用や使い分けの判断へつなげます。二候補の A/B テストと、複数候補のマルチアーム検証に同じ骨格を使います。

比較する選択肢に応じて、採用・使い分けの判断に使う結果をそろえます。次の表は、UI、処理ロジック、AI 構成で用いる比較軸です。

表 2: 選択肢ごとに比較する結果

Arm として変えるもの	**比較する結果**
画面・操作方法	操作完了率、所要時間、ユーザーの評価。
処理ロジック	正しさ、レイテンシ、スループット、資源消費。
AI の構成	生成結果の品質、AI コスト、処理時間。

AI 構成では、各 Arm・ケースに、生成結果の品質評価と、その結果を得るまでの AI コストを対で残します。品質は、人が修正する前の生成結果を、正確性・情報の抜け・業務条件への適合など、同じ基準で評価します。生成・AI による確認・再試行のコストを集計し、人による確認・修正の負担と処理時間も併記します。

評価は、実際の利用を通じた比較と、Corpus を使って同じ条件から繰り返す比較の両方で行えます。A/B・マルチアームは候補の数や比較構成を表し、Corpus は入力ケースと期待する結果・根拠を固定します。Corpus を使う検証でも、A/B・マルチアームで候補を比較できます。

図 3 は、同じ Corpus から Experiment の各 Arm を実行し、生成結果の品質と AI コストを組にして比較する記録へそろえる構造です。

8.1 AI チューニングとコスト削減の根拠を残す

AI の総費用は単価だけでなく、配置した工程や再試行・修正にも左右されます。

AI チューニングでは、モデル・設定・工程配置ごとに AI コストと生成結果の品質を組にして評価し、必要な品質を満たす構成の中から、費用と性能に合う候補を探します。

複数社のモデル、ローカル LLM、リーズニングレベルを組み合わせて Arm を作ります。同じ Corpus と品質基準で、各 Arm の生成結果と、生成・確認・再試行を含む AI コストを対応付けます。人の修正で生成品質の差が見えなくならないよう、修正前の品質評価と確認・修正の負担を分けて残します。

8.2 Corpus を基盤に比較と再検証を繰り返す

AI チューニングで構成と入力を同時に変えると、結果の差を切り分けにくくなります。

Corpus の版・ケースと比較条件を固定し、各 Arm を同じ起点から繰り返し実行します。Corpus を比較の基盤にすることで、構成の違いを検討する根拠を積み重ねます。

版・ケース・Arm・実行と品質・性能・費用を対応付けます。新しいモデルやリーズニングレベル、複数社・ローカル LLM の組み合わせでも同じケースを使います。修正や失敗事例は次の Corpus の材料とし、固定版での比較と評価ケースの充実を分けて進めます。

9 課題ごとの対応

性能の変化、余分な処理、AI 費用の増加では、原因を調べるために必要な記録の組み合わせが異なります。

時間と件数、呼び出しと資源・進捗、条件と消費の内訳を、それぞれの課題へ対応付けます。

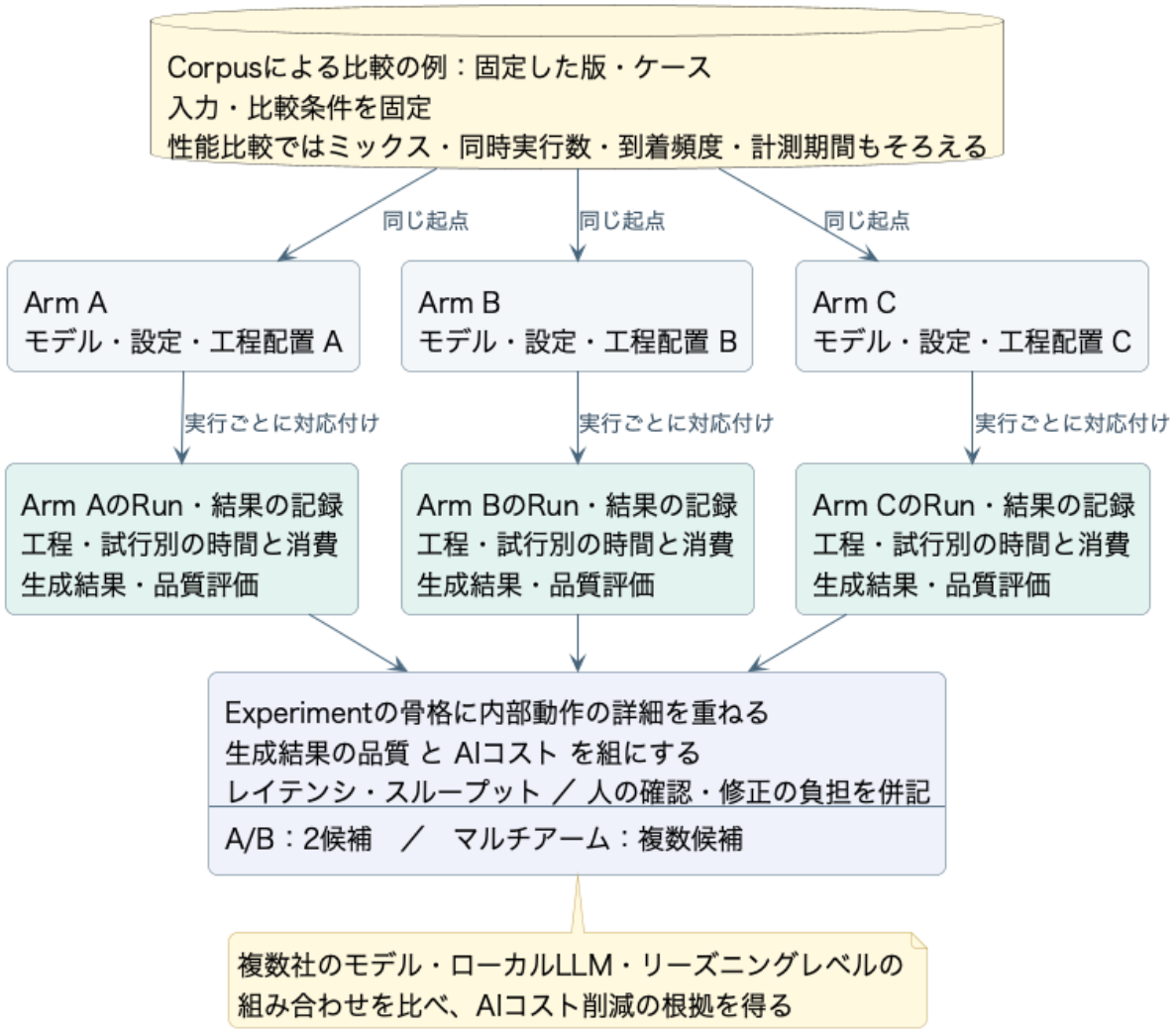

図 3　同じ Corpus から各 Arm の比較記録をそろえる

9.1　性能を処理区間と利用条件へ対応付ける

要求群や同時実行条件が異なると、レイテンシやスループットも変わります。

要求の時間と完了件数を、リクエストミックス・実行条件へ対応付け、性能を比較可能な実績にします。

Corpus の版・ミックス、同時実行数・到着頻度・計測期間を残します。完了件数を期間の長さで割って全体・種類別のスループットを求め、要求ごとのレイテンシと合わせて、同じ条件の改善前後や候補を比較します。

9.2　負荷を呼び出しの回数・構造と合わせて捉える

CPU 負荷の上昇だけでは、負担を生んだ呼び出しを特定できません。

同じ実行の呼び出し構造・回数と、資源消費・進捗を合わせ、重複、I/O 反復、再試行や暴走の箇所を調べます。

Operation や I/O の時間に、その実行を含むプロセス・期間の CPU 負荷やキャッシュの計測を範囲付きで関連付けます。消費の累積と進捗も読み、入力件数に比例した高コストな I/O や、進捗に見合わない消費を調べます。

9.3 AI の呼び出し・時間・費用を実行へ結び付ける

AI 費用の合計が増えたときは、モデル変更、不要な呼び出し、再試行の影響を分けて調べる必要があります。

AI の条件・構成・試行へ時間と消費を対応付け、動作の問題と構成による負担を切り分けます。

呼び出し元・入力条件・工程・AI 構成と時間・利用量・費用を Run と実行結果の記録へ対応付けます。呼び出し構造から工程をたどり、どこで AI コストが生じたかを読み取ります。候補比較では Arm と生成結果の品質評価をそろえ、Corpus を使う場合は版・ケースも加えて、AI コストと品質を組で評価します。

9.4 観測対象と必要な記録の一覧

各課題の分析を組み立てる際は、捉えたい現象に必要な記録を選ぶ必要があります。

性能には時間と件数、負荷には呼び出しと資源消費、AI には条件・構成と利用量を対応付けます。次の表で、捉えたいことと必要な記録の違いを比較します。

表 3: 観測対象・捉えたいこと・必要な記録

観測対象	捉えたいこと	必要な記録
性能	一件の速さ、リクエストミックスごとの処理量、時間を使う区間。	開始・終了・区間ごとの時間、全体・種類別の完了件数、Corpus の版・ミックス、同時実行数・到着頻度・計測期間。
負荷と無駄な処理	資源消費、重複・反復・再試行、進捗に対する消費の増え方。	呼び出し構造・回数、CPU・I/O・キャッシュの計測、再試行・進捗・累積量。
AI の動作と費用	意図しない呼び出し、長時間の実行、工程・モデル・再試行別の消費。	呼び出し元・入力条件・開始終了・時間、工程・AI 構成・試行、利用量・費用と算定根拠、結果・評価への参照。

10 観測モデルを組み立てて提示する

一件の遅延を調べる場合と、一定期間の処理量を比べる場合では、必要な記録の範囲や粒度が異なります。

分析する問いに合う範囲で観測モデルを作り、人間と AI へ同じ根拠を提示します。モデルとの照合を行う用途では、要求モデルと設計モデルへの対応をそれぞれ用意します。

10.1　対象と粒度を選んで記録をまとめる

記録の全量を常に同じ細かさでモデル化すると、分析対象と変換の負担が大きくなります。

着目するアプリケーション・処理・期間を選び、問いに合う粒度で記録をまとめます。

選んだ範囲の自動トレースを共通処理へ渡し、実行 ID・Operation・親子関係・開始終了を定型の表現へ整理します。入力・条件・Arm・Run・結果・評価の骨格へ、呼び出し構造やその他のログ・計測値を対応付け、元の記録への参照を保持します。問いに合う対象と粒度を選び、必要な詳細をたどります。

10.2　人間と AI に同じ実行を提示する

図で見つけた偏りを AI へ調べさせるには、対象実行と計測範囲が一致している必要があります。

人間向けの図・表と AI 向けの構造化データを同じ識別子へ結び付け、同じ実行の根拠を共有します。

図や表に構造・回数・時間と、対象の処理・期間・条件を示します。AI へも同じ実行・呼び出しの識別子と計測範囲を渡し、人間が着目した記録を具体的な分析の対象にします。

10.3　モデルとの照合には対応付けを加える

Operation とモデル要素では、役割も識別子も異なります。実行の記録だけから、モデル上のどの要素かを一意に読み取ることはできません。

要求モデルと設計モデルへの対応を別々に用意し、実行の事実をそれぞれのモデルと照合します。

10.3.1　ユースケースとの対応を用意する

要求モデルでは、ユースケースを軸に業務上の期待を表します。実際の操作列と比べるには、期待を実装に近い操作列まで具体化する必要があります。

ユースケースの操作と Operation を対応付け、期待する操作列と実際の呼び出し列を比較できるようにします。

モデル上の操作・Operation・対象実行を識別できる対応を自動トレースへ加えます。この対応を用いて、要求モデルの期待する列と観測した列を別々の対象として並べ、検証に使う根拠を用意します。

10.3.2　設計モデルの処理単位への対応を用意する

時間を使った Operation が分かっても、設計上のどのコンポーネントや処理単位に属するかは、別途対応付ける必要があります。

Operation や計測区間を設計要素へ対応付け、負担をコンポーネントや処理単位で分析します。

設計モデルは実現構造、要求モデルは業務上の期待を表します。それぞれへの対応を別に持つことで、同じ実行の事実を、設計の改善と要求の確認という異なる問いへ使えます。

11　性能・負荷・AI コストと比較検証の記録例

調査や構成比較では、入力・条件と、結果に至る呼び出し・負担を結び付けた記録を読み取ります。

以下の架空の例で、性能、I/O の反復、AI 費用、Arm 別の比較について、観測モデルが用意する根拠を具体化します。

11.1　同じリクエストミックスで性能改善を比べる

架空の帳票システムで、帳票生成の性能を改善し、一覧照会への影響も確かめる場合を考えます。

同じリクエストミックスで変更前後を実行すると、処理量の改善と、各種類の要求への影響を比較できます。

Corpus に一覧照会 70%、帳票生成 30% と、取得件数・明細数を定義します。同じ版・ミックス、同時実行数・到着頻度・計測期間で実行し、要求ごとの時間、全体・種類別の完了件数、資源消費を残します。検索・集計・出力の呼び出しと合わせ、性能が変わった区間を読み取ります。

11.2　1000 個の明細の I/O 反復の例

1000 個の明細を処理する仮想例として、明細ごとの照会で同種の外部 I/O を 1000 回呼ぶ場合を考えます。出力が正しくても、呼び出しの反復が時間と資源を使います。

観測モデルからバルク I/O の候補を見つけ、処理を改修し、同じ Corpus で変更前後を検証します。候補の発見から改善の確認まで、次の三段階でつなげます。

- バルク I/O の候補を見つける
- バルク I/O へ改修する
- 同じ Corpus で変更前後を検証する

候補を見つける段階では、CallTree で同種の I/O が繰り返される箇所を特定し、入力 1000 個、I/O 回数、処理区間の時間を対応付けます。まとめて処理しても出力と業務上の条件を保てる箇所を、バルク I/O の候補にします。

改修では、明細ごとに行っていた照会をまとめ、1000 個を一回で取得するバルク I/O へ変更します。取得結果を明細へ対応付ける処理と、出力の意味を保ちます。

検証では、改修前の処理とバルク I/O 版を Arm として定義します。同じ 1000 個の入力と期待する出力を持つ Corpus で両方を Run として実行し、各 Run に結果・計測値・評価を対応付けます。出力の一致、I/O 回数の 1000 回から 1 回への変化、レイテンシ・資源消費を比べ、必要に応じて同じリクエストミックスのスループットも比較します。

図 4 は、反復する I/O の候補発見から、バルク I/O への改修、同じ Corpus を使った変更前後の検証までを示します。

11.3　AI の再試行を含むコスト内訳を表す

架空の注文書抽出で、AI を再試行した後に結果を採用した場合を考えます。

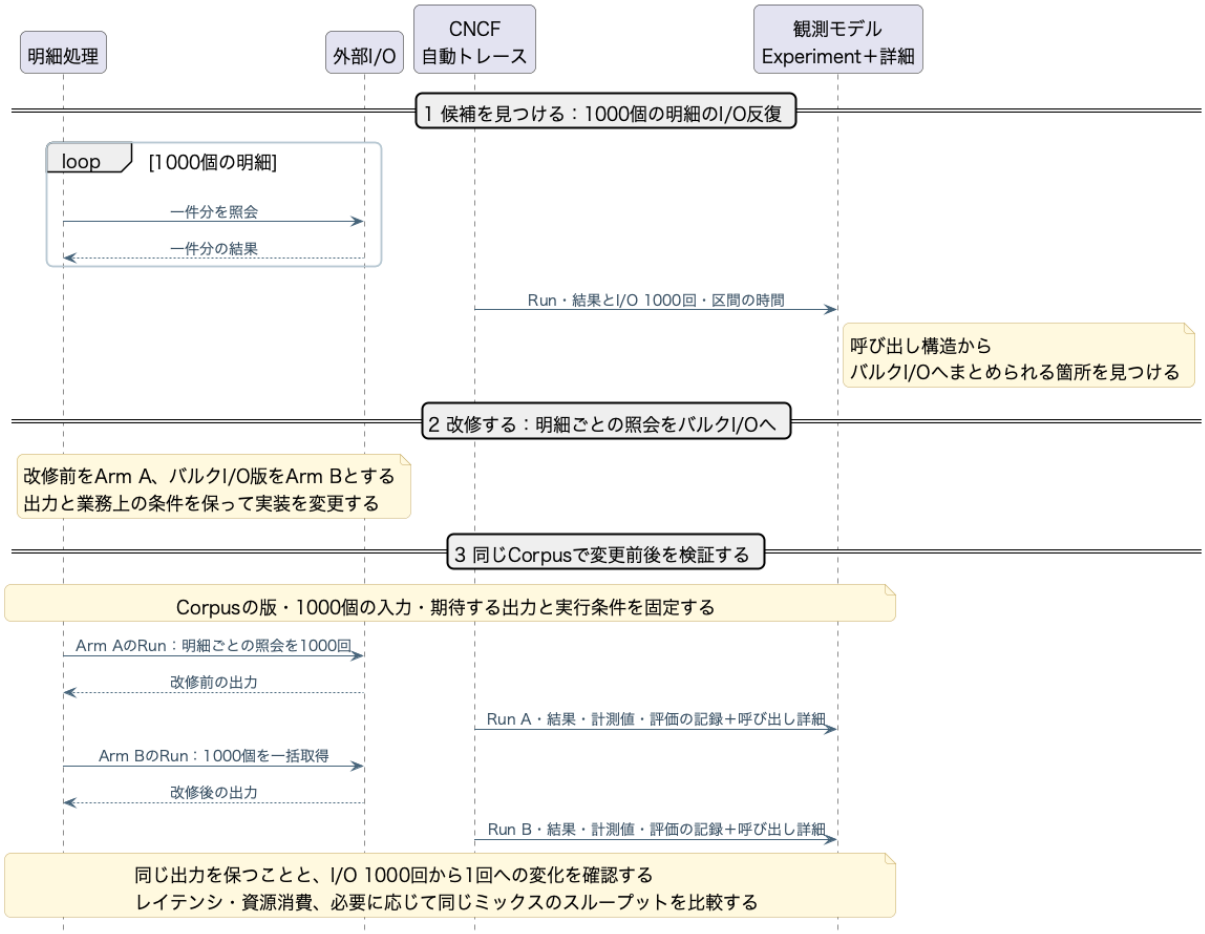

図 4　候補発見・バルク I/O への改修・Corpus による検証

試行別の消費を最終結果と結び付けることで、再試行が品質と総費用へどう関係したかを調べられます。

各試行のモデル・版・リーズニングレベルと時間・利用量・AI コストを残し、Run と実行結果の記録に出力への品質評価と編集者による修正・判断を対応付けます。試行別・工程別の消費から、採用する生成結果を得るまでの AI コストを読み取ります。

11.4　同じ注文書を処理した Arm 別の実績を表す

架空の注文書抽出で、Arm A はクラウド AI、Arm B はローカル LLM の抽出とクラウド AI の確認を組み合わせるとします。

共通の Corpus ケースで、Arm ごとに AI コストと生成結果の品質を組にして残すと、工程構成の違いを判断できます。

同じ Corpus の版・注文書を各 Arm で Run として実行し、人が修正する前の生成結果を、項目の正確性や抽出漏れなど共通の基準で評価します。その評価と、抽出・AI による確認・再試行を含む AI コストを、各 Run に対応する記録として残します。処理時間と人による確認・修正の負担も併記し、必要な生成品質を満たす候補の費用と性能を比べます。

12　まとめ

観測モデルは、Experiment を骨格に内部動作の解析に必要な詳細を加え、不具合や性能の調査と、選択肢の比較を一つのモデルでつなげます。

その役割を支える要点は、次の三つです。

- 開発時の検査に加え、実行時の性能・負荷・AI 利用を捉えます。二重実行、過剰な I/O、意図しない AI 呼び出しや費用増加を検証する根拠を残します。
- CNCF の自動トレースを使って呼び出し構造や I/O 回数などの詳細を整え、アプリ側の観測用追加実装なく分析へつなげます。可視化によって人間の直感を生かし、議論の文脈を共有できます。
- 実利用での比較と Corpus による再実行を、入力・条件・Arm・Run・結果・評価の対応で表します。バルク I/O 改修前後の検証と、生成品質・AI コストによる AI 構成の比較を、同じ枠組みで繰り返せます。

観測モデルは、実行の事実を捉えることでハーネスを支えます。この根拠を用いた基準との照合と改善へのつなぎ方は、次回の検証ハーネスで扱います。