観測モデル

Experimentを骨格に、内部動作の詳細を加える

不具合の検出と調査

実際の動作と負担をたどる

よりよい構成の探索

品質・性能・費用を比べる

Experimentの骨格と内部動作の詳細

観測モデル ＝ Experiment ＋ α

Experiment：骨格

入力・条件・Arm・Run
結果・計測値・評価

＋

α：内部動作の詳細

呼び出し・I/O回数・時間
CPU・キャッシュ・AI利用

同じRunへ結び付けて読む

不具合の調査 ／ よりよい構成の探索

入力・条件・Arm・Run・結果・評価と、呼び出し・時間・資源を同じRunへ結び付ける。

モデル化によって分析を繰り返し使える

解析を定型化	生のログを毎回変形せず、同じ形を解析
可視化で理解と議論	人間の直感を生かし、共通の文脈で議論
Corpusで再利用	入力・比較条件を保存し、同じ起点で再実行

モデル化の意義は、二つの利用目的を共通に支える。

性能・負荷・AI費用から動作を捉える

性能	レイテンシとスループットを条件別に記録
負荷と無駄な処理	CPU、I/O、重複、反復、キャッシュを観測
AIの動作と費用	意図しない呼び出し、長時間実行、再試行を追跡

開発時の検査で捉えきれない現象を、実際の動作から調べる。

実利用の観測とCorpus再実行

実利用の観測

入力・条件・Armの割当てと結果を保存

実際に起きたことを分析・比較

Corpusによる再実行

入力ケースと期待する結果・根拠を版で固定

同じ起点から新しいRunで繰り返し比較

A/B・マルチアームは、実利用とCorpus再実行の両方で使う比較の構成。

Corpusでリクエストミックスを定義する

要求の組み合わせ	種類・割合・入力規模をCorpusに定義
実行条件	同時実行数・到着頻度・計測期間をそろえる
性能の実績	全体・種類別のスループットとレイテンシを残す

同じミックスと条件を使い、性能改善の効果を繰り返し比べる。

生成品質とAIコストを組にして比べる

Armの構成

複数社のモデル・ローカルLLM
推論設定と工程配置を組み合わせる

同じCorpusで比較

人の修正前の生成品質を評価
生成・AI確認・再試行のコストと対にする

採用・使い分け

必要な品質を満たす候補の費用と性能を比較
人による確認・修正の負担は別に記録

ExperimentのRunに、結果・計測値・評価と内部動作の詳細を対応付ける。

1000個の明細のI/O反復の例

候補を見つける

CallTreeで1000回のI/Oと区間の時間を確認
出力を保って一括化できる箇所を選ぶ

バルクI/Oへ改修

1000個をまとめて取得する実装へ変更
改修前後をArm A・Arm Bとして定義

同じCorpusで検証

Runごとの出力・I/O回数・時間・資源を比較
同じミックスでスループットも確認

候補の発見 → 実装の改修 → 同条件での再実行・比較までをつなげる。

観測モデル

Experimentを骨格に、内部動作の詳細を結び付ける

観測モデル ＝ Experiment ＋ 内部動作の詳細（α）

Experiment：比較・再実行の骨格

入力・実行条件・Arm・Run
Runに対応する結果・計測値・評価
候補を定義 → 割当て → 実行・評価 → 選択へ反映

α：動作解析のための詳細

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・AI利用量と費用
CNCFの自動トレースと各計測を対応付ける

同じRunへ結び付け、結果の差と内部の動作を一緒に読む

不具合の検出と調査

重複実行・I/O反復・余分なAI呼び出しを調べる

よりよい構成の探索

処理方式・UI・AI構成の品質、性能、費用を比べる

実利用の観測

入力・Armの割当て・結果を記録し、分析・比較

Corpusによる再実行

同じ版・ケースから繰り返し実行し、比較

解析に適した定型化 ／ 可視化で直感と議論を支える ／ Corpus化で反復利用

CNCFの共通基盤で整備・分析し、アプリ側の観測用個別実装を省く

観測モデルがハーネスを支える

共通の根拠	Experiment＋内部動作の詳細で調査と探索をつなぐ
アプリの観測用追加実装を省く	CNCFの自動トレースと共通のモデル化・分析を使う
分析と判断に再利用	可視化で直感と議論を支え、Corpusで比較を繰り返す

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へ。