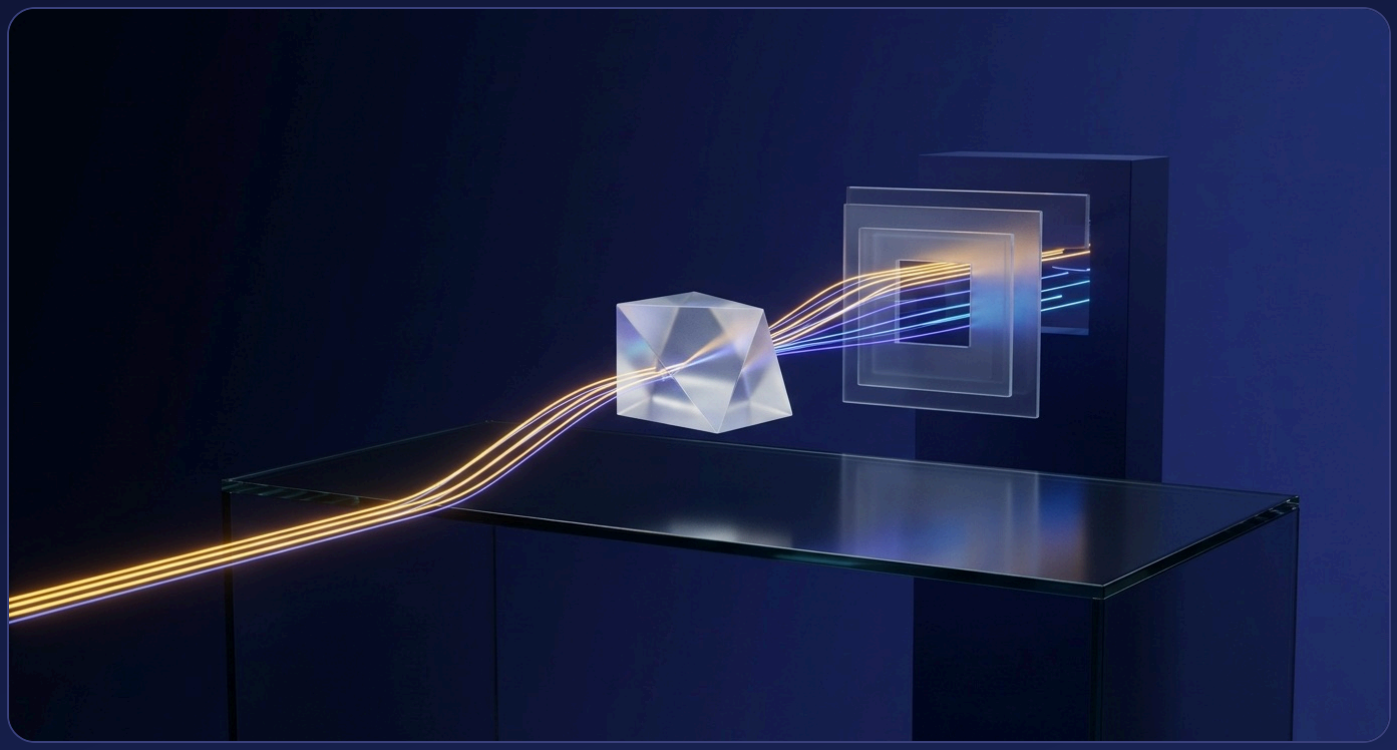
AI AGENT ENGINEERING & FINOPS • FIELD GUIDE

Agentic Observability & Prompt Caching

The complete 12-page visual handbook to OpenTelemetry GenAI trace trees, static-first prefix caching, historical tool compaction, dynamic model routing, and production token economics.

OTel gen_ai.* Spans | Cache Breakpoints & TTLs | Tool-Output Compaction | Subagent Trace Trees | Cheap-to-Frontier Routing | Eval-Linked FinOps



Up to 90% Cache Savings
Reuse stable system instructions, tool schemas, and frozen turns at 0.1x read pricing instead of paying full input rates every step.

Unified Multi-Agent Traces
Carry W3C traceparent context across subagent handoffs and MCP servers so nested calls appear in one clean waterfall.

Eval-Linked Cost Control
Track time to first token, reasoning tokens, and cost per task side by side with Langfuse or Arize AX evaluation scores.

The Telemetry & Caching Mental Model: Trace First, Cache the Prefix

Because multi-turn agents resend their full conversation history on every step, uncached agent loops create quadratic token bills. Combining OpenTelemetry GenAI spans with prefix caching keeps unit economics predictable and failures transparent.

Quick Navigation

1. The Telemetry & Caching Mental Model p. 2
2. Provider Cache & OTel Matrix p. 3
3. Annotated Trace Waterfall & Cache HUD p. 4
4. OTel GenAI Spans & Subagent Propagation p. 5
5. Static-First Prefix & Cache Breakpoints p. 6
6. Compaction & Dynamic Model Routing p. 7
7. Breakpoints, Compaction & Ledger Code p. 8
8. Five Copy-Paste Production Recipes p. 9
9. PII Redaction, Alerts & FinOps Rollout p. 10
10. End-to-End Synergy Pipeline p. 11
11. Troubleshooting Clinic & Pre-Flight QA p. 12

T OTel Trace Hierarchy

Structure each workflow as a parent `invoke_agent` span wrapping child `chat` model calls, `execute_tool` spans, and linked subagent trees.

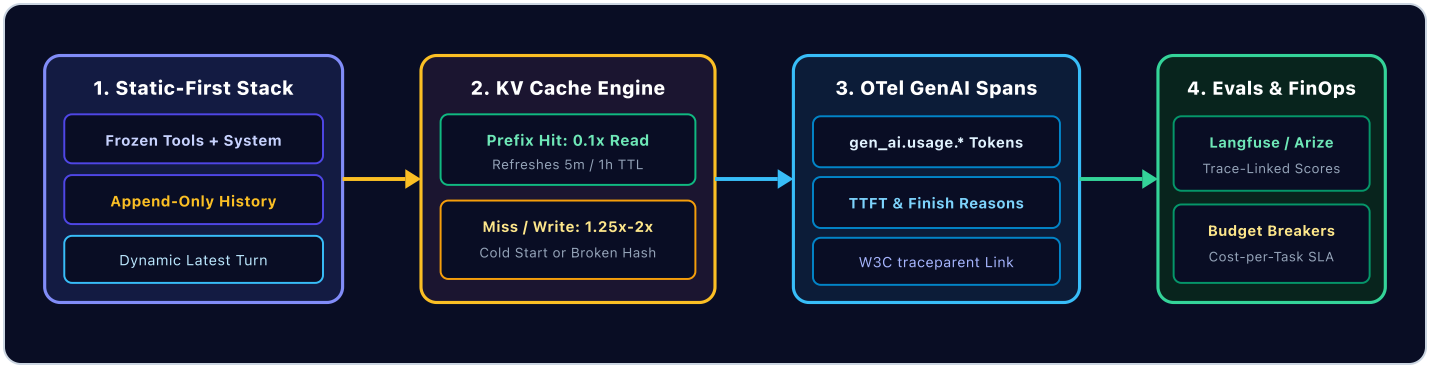
C Append-Only Prefix Stack

Lock unchanging tool schemas and system instructions at the very top of every request so provider KV caches stay warm across turns.

R Compaction & Routing

Trim bulky historical tool payloads in scheduled batches and route simple classification steps to fast, inexpensive model tiers.

Architecture: Request Stack → Provider KV Cache → OTel Exporter → Eval Ledger



Provider Prompt Caching & OTel Telemetry Matrix

Every major model provider handles prefix reuse differently. Knowing minimum token floors, write multipliers, and TTL timers lets you avoid silent cache misses and map exact usage fields into OpenTelemetry.

Provider Mode	Activation & Threshold	Write & Read Pricing	Default & Max TTL	Usage Telemetry Field
Anthropic Cache Explicit & Auto	Up to 4 `cache_control` breakpoints (`ephemeral`) or top-level auto flag (≥1,024–2,048 tok)	5m write: `1.25x` 1h write: `2.0x` Read: `0.1x` (90% off)	5 min default or `ttl: "1h"`; resets on every hit	`cache_creation_input_tokens` `cache_read_input_tokens`
OpenAI Prefix Auto + Options	Automatic ≥ 1,024 tokens; newer models support `prompt_cache_options`	Zero auto write fee; 90%–95% read cut; 12h tier write fee	30 min default; up to 12h extended retention	`prompt_tokens_details.cached_tokens`
Gemini Implicit Gemini 2.5 / 3 / 4	Always on by default (min 2,048 tok on 2.5; 4,096 tok on 3.x / 4.x)	Zero storage fee; ~90% hit discount (best-effort)	Auto-managed (up to ~24h Cloud window)	`usage_metadata.cached_content_token_count`
Gemini Explicit CachedContent API	Manual `CachedContent` resource created ahead of agent sessions	Standard write + hourly rent (~$1/1M tok/hr) + ~90% cut	Developer-set TTL (guaranteed hold)	`cached_content_token_count` + cache ID

OpenTelemetry Normalization & The Break-Even Rule

1 Migrating Legacy `gen_ai.system`

OpenTelemetry GenAI conventions carry Development status and replace `gen_ai.system` with `gen_ai.provider.name` (`anthropic`, `openai`, `gcp.vertex_ai`).

Set `OTEL_SEMCONV_STABILITY_OPT_IN=gen_ai_latest_experimental` and normalize attributes inside your Collector.

2 Cache Write Break-Even Math

A 5-minute write (`1.25x`) breaks even on **1 hit** (`1.35x` vs `2.0x` raw). Never write a cache read zero times (+25% waste).

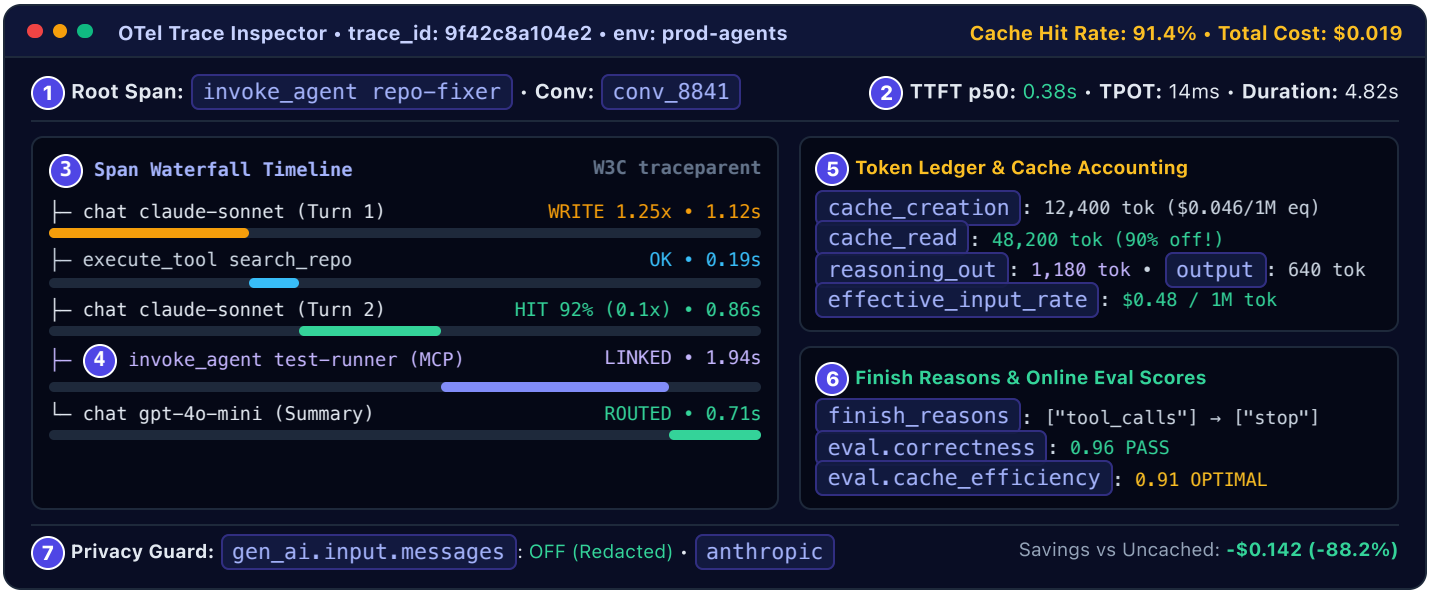
A 1-hour write (`2.0x`) breaks even on **2 hits** (`2.2x` vs `3.0x`)—ideal whenever pauses exceed 5 minutes.

Always Track Total Input vs. Cache Read Separately

In OTel GenAI spans, `gen_ai.usage.input_tokens` includes cached tokens, while `gen_ai.usage.cache_read.input_tokens` and `gen_ai.usage.cache_creation.input_tokens` break out discounted and write-premium counts for true FinOps billing.

Annotated Workspace: Trace Waterfall & Cache-Hit HUD

Modern tracing dashboards such as Langfuse and Arize AX combine distributed OpenTelemetry timelines with token ledgers and online eval scores so you can diagnose slow turns or cache misses in seconds.



Control Legend & Inspection Points

1 **Root Agent Span:** Identifies the top-level workflow via `invoke_agent` and ties multi-turn episodes together with `gen_ai.conversation.id`.

2 **Streaming Latency Gauges:** Tracks `gen_ai.server.time_to_first_token` (TTFT) and time per output token (TPOT) to separate cold prefill delays from generation speed.

3 **Span Waterfall Timeline:** Visualizes Turn 1 cache creation (amber), tool gaps, Turn 2 cache hits (green), and dynamically routed summary calls.

4 **Subagent Handoff Link:** Connects child `invoke_agent` trees triggered over async queues or MCP servers via propagated W3C `traceparent` headers.

5 **Granular Token Ledger:** Separates cache writes, cache reads, reasoning tokens, and final output tokens to compute true effective input pricing.

6 **Finish Reasons & Evals:** Verifies clean `tool_calls` to `stop` progression (flagging `length` cutoffs) alongside trace-linked Langfuse or Arize AX eval scores.

7 **Content & PII Guard:** Confirms prompt bodies stay disabled or scrubbed by an OTel Collector redaction processor before leaving your secure VPC.

OpenTelemetry GenAI Spans, Metrics & Subagent Propagation

Vendor-neutral `gen_ai.*` spans let your agents switch between Anthropic, OpenAI, and Gemini without rewriting telemetry pipelines, while W3C trace propagation keeps distributed subagent trees intact.

1 `invoke_agent`

Marks agent or subagent boundaries with `gen_ai.agent.name`, `gen_ai.agent.id`, and `gen_ai.conversation.id`.

2 `chat` Model Spans

Records `gen_ai.usage.*` token attributes, `gen_ai.response.finish_reasons` (`stop`, `tool_calls`, `length`), and TTFT histograms.

3 `execute_tool`

Wraps tool calls and MCP requests with `gen_ai.tool.name` and `gen_ai.tool.call.id` to isolate external latency.

Propagating Context Across Subagent & MCP Boundaries

When an orchestrator delegates work over an HTTP webhook, task queue, or Model Context Protocol (MCP) tool call, automatic thread-local context is lost. Always inject the active W3C `traceparent` into payload metadata (such as MCP `_meta`) and extract it before creating child spans, or derive deterministic trace IDs from a shared correlation key.

```
// TypeScript: Register Vercel AI SDK Telemetry + Subagent MCP Trace Propagation
import { registerTelemetry } from "ai";
import { context, propagation } from "@opentelemetry/api";
import { LangfuseSpanProcessor } from "@langfuse/otel";
import { LangfuseVercelAiSdkIntegration } from "@langfuse/vercel-ai-sdk";

// 1. Export standardized gen_ai.* spans to Langfuse or any OTLP backend
export function register() {
  registerTelemetry({ integrations: [new LangfuseVercelAiSdkIntegration()] });
}

// 2. Inject W3C traceparent into MCP _meta before handing off to a subagent
export function buildMcpHandoffParams(toolArgs: Record<string, unknown>) {
  const carrier: Record<string, string> = {};
  propagation.inject(context.active(), carrier);
  return { ...toolArgs, _meta: { traceparent: carrier.traceparent, tracestate: carrier.tracestate } };
}
```

Attaching Trace-Linked Evaluation Scores (Langfuse & Arize AX)

Tracing only tells you what happened; evaluation scores tell you whether it worked. Both Langfuse and Arize AX ingest `gen_ai.*` or OpenInference spans and let online evaluators write correctness, groundedness, and cache-efficiency scores directly back onto each `trace_id`.

```
# Python: Record Trace-Linked Online Eval Scores Back to Trace ID
eval_payload = {
    "trace_id": span_context.trace_id_hex,
    "scores": [
        {"name": "task_correctness", "value": 0.96, "comment": "All unit tests passed after patch"},
        {"name": "cache_hit_ratio", "value": cache_read_tokens / max(total_input_tokens, 1)},
        {"name": "doom_loop_detected", "value": 0, "data_type": "BOOLEAN"}
    ]
}
observability_client.score_trace(**eval_payload)
```

Key Metric Instruments to Dashboard First

Pair span attributes with OpenTelemetry metric histograms: `gen_ai.server.time_to_first_token` (p95 cold prefill vs warm cache), `gen_ai.server.time_per_output_token` (streaming throughput), and `gen_ai.client.token.usage` broken down by token type (`input`, `output`, `cache_read`, `reasoning`).

Static-First Prefix Ordering & Cache Breakpoint Architecture

Provider KV caches match prompts strictly from byte zero forward. One dynamic timestamp or reordered tool schema near the top of your payload invalidates every downstream token on every turn.

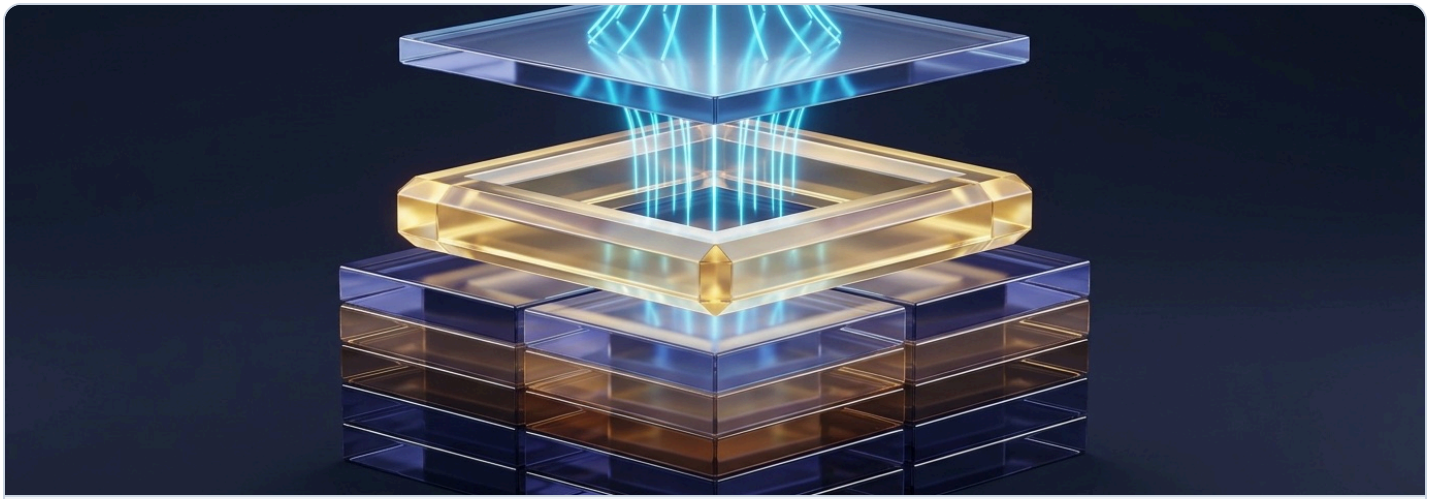


Figure 6.1 • Immutable Prefix Stack: Foundational tools, system rules, and frozen history stay locked inside the golden KV cache base, while only new delta tokens pay full compute on each turn.

Anatomy of a Cache-Friendly Prompt Stack (Tools → System → History → Turn)

[1. Tools Array (Frozen)] All JSON schemas sorted deterministically; never add or remove tools mid-run.
[2. System Prompt + Breakpoint #1] Static persona, policies, repo conventions → `cache_control: ephemeral`.
[3. Append-Only History + Breakpoint #2] Immutable user/assistant/tool turns frozen up to the prior step.
[4. Volatile Latest Turn (Uncached)] Current time, live git status, active file diff, and user prompt go strictly last.

1 Three Silent Cache Killers

Top-of-Prompt Timestamps: Putting `Current time: 14:32:07` inside the system prompt forces a 0% cache hit rate. Push timestamps into the newest user message or expose a `get_time` tool.

Dynamic Tool Swapping: Dropping unavailable tools invalidates the Tools tier and everything below it. Keep the tool array static and restrict calls via `tool_choice` or policy validation.

2 Choosing 5-Minute vs. 1-Hour TTL

Default 5-Minute TTL (`1.25x` write): Optimal for tight autonomous loops where tool executions finish in seconds and turns fire continuously.

Extended 1-Hour TTL (`2.0x` write): Essential for human-in-the-loop coding assistants, code review approvals, or slow CI waits that often pause for 6 to 45 minutes between turns.

Tool-Output Compaction & Dynamic Cheap-to-Frontier Routing

Long-horizon agents quickly accumulate tens of thousands of stale search logs and file dumps. Compaction keeps working memory crisp, while dynamic routing protects margins from verbose reasoning output costs.

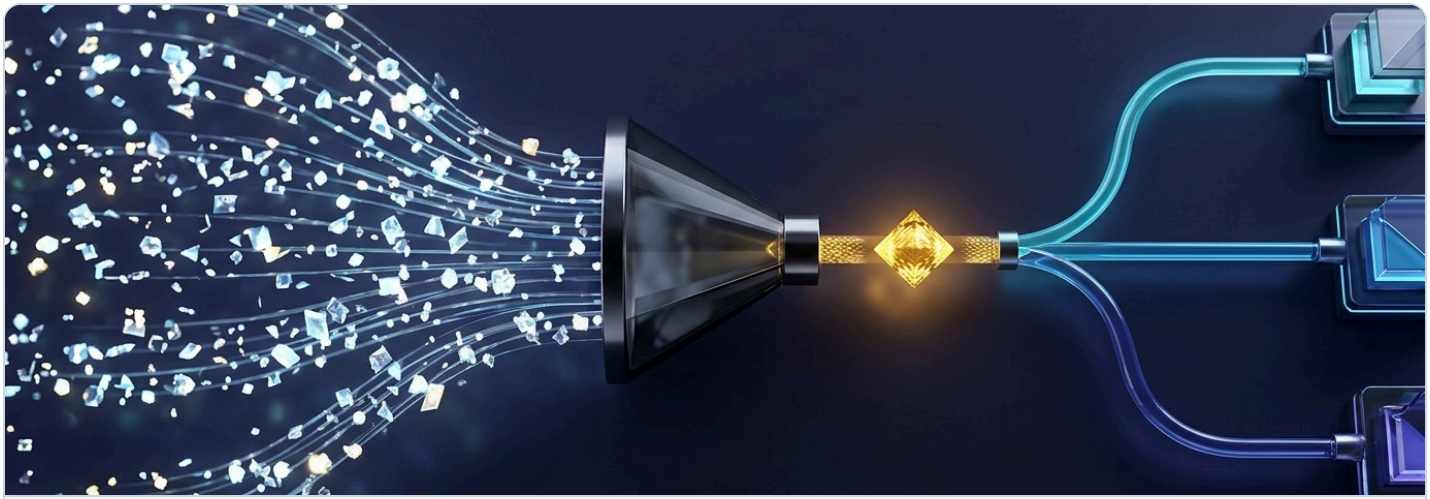


Figure 7.1 • Compaction Funnel & Model Cascade: Verbose historical tool outputs pass through a pruning gate into condensed summaries, then route across fast small models and frontier reasoning tiers.

1 Historical Tool-Output Compaction

Observation Truncation: Cap raw bash, log, or DOM tool responses immediately (keeping head/tail slices or structured JSON fields) before appending to history.

Server-Side Compaction: Use Anthropic context editing (`clear_tool_uses` / `compact-2026-01-12`) or OpenAI Responses API `compact_threshold` (which returns encrypted compaction items).

Batch Rule: Never compact on every turn—rewriting history resets the KV cache prefix. Compact in milestone batches (for example, once history passes 60K tokens).

2 Cheap-to-Frontier Model Routing

Output Cost Asymmetry: Generated output and reasoning tokens cost 3x to 8x more than standard input tokens—and up to 50x more than cached reads.

Tiered Cascade: Route intent routing, JSON formatting, log summarization, and guardrail evals to fast small models (such as Haiku, Flash, or mini tiers); escalate architecture and tricky bug fixes to frontier reasoning models.

Subagent Isolation: Offload noisy exploration to cheaper worker subagents that return a crisp 1,500-token synthesis to the main cached orchestrator.

Routing + Caching Synergy: Isolate Each Model Tier's Prefix

Prompt caches are keyed per model ID, so never flip the same growing conversation array back and forth between a small model and a frontier model. Instead, route side-tasks (classification, formatting, compaction summaries) into isolated single-turn calls or dedicated small-model subagents while keeping the main reasoning prefix locked on one primary model.

Implementing Cache Breakpoints, Compaction & Cost Ledgers

Translate prefix rules into explicit provider payloads and build a ledger-accurate cost calculator that weights cache writes, 0.1x cache reads, reasoning tokens, and loop circuit breakers.

1. Anthropic Breakpoints (`1h` TTL) & Server-Side Context Editing

```
{
  "model": "claude-sonnet-4-6",
  "tools": [ /* Frozen sorted schemas; breakpoint placed on final tool */ ],
  "system": [{
    "type": "text",
    "text": "You are a senior reliability engineer...",
    "cache_control": { "type": "ephemeral", "ttl": "1h" }
  }],
  "context_management": {
    "edits": [{ "type": "clear_tool_uses_20250919", "trigger": { "type": "input_tokens", "value": 50000 }, "keep":
{ "type": "tool_uses", "value": 4 } }]
  },
  "messages": [ /* Append-only turns; mark penultimate turn with cache_control */ ]
}
```

2. OpenAI Responses Compaction & Gemini Explicit CachedContent

```
// OpenAI Responses API: Automatic server-side compaction at 64k tokens
const response = await openai.responses.create({
  model: "gpt-5",
  input: conversationItems,
  context_management: [{ type: "compaction", compact_threshold: 64000 }]
});

// Gemini Explicit Caching: Guarantee 90% discount on large static repository bundles
const cache = await ai.caches.create({
  model: "gemini-2.5-pro",
  config: { contents: [repoBundlePart], ttl: "3600s", systemInstruction: staticRules }
});
```

3. Ledger-Accurate Effective Input Price & Doom-Loop Guard

```
def compute_turn_cost_and_guard(usage, base_in_per_m, base_out_per_m, ttl="5m"):
    write_mult = 2.0 if ttl == "1h" else 1.25
    uncached = max(usage.input_tokens - usage.cache_read - usage.cache_creation, 0)
    in_cost = (uncached + write_mult * usage.cache_creation + 0.10 * usage.cache_read) * (base_in_per_m / 1e6)
    out_cost = (usage.output_tokens + usage.reasoning_tokens) * (base_out_per_m / 1e6)
    effective_in_rate = (in_cost / max(usage.input_tokens, 1)) * 1e6
    return {"total_usd": round(in_cost + out_cost, 6), "effective_in_per_m": round(effective_in_rate, 3)}
```

Measuring True Cache Health with Effective Input Rate

Comparing `effective_in_per_m` against your model's base input rate exposes cache regressions immediately: a healthy multi-turn agent loop drops effective input cost by 75% to 88% below list price even after accounting for initial `1.25x` writes.

Five Copy-Paste Recipes for Observable, Cached Agents

Ready-to-deploy patterns covering framework telemetry hooks, multi-provider cache breakpoints, explicit context bundles, subagent span stitching, and OpenTelemetry Collector normalization.

Recipe & Goal	Stack & SDK	Core Configuration Pattern	Observability / FinOps Win
1. Next.js AI SDK Tracing Stream spans + evals	Vercel AI SDK + Langfuse OTel	`registerTelemetry({ integrations: [new LangfuseVercelAiSdkIntegration()] })`	Captures TTFT, finish reasons, and nested tool spans out of the box.
2. Coding Assistant Cache Survive human pauses	Anthropic Messages API	`cache_control: {"type": "ephemeral", "ttl": "1h"}` on tools & system	Eliminates 5m TTL expirations during manual diffs; cuts read cost 90%.
3. Repo-Scale Gemini Cache Lock 100K+ context	Google GenAI SDK	`ai.caches.create({ contents, ttl: "3600s" })` (above 4,096 token floor)	Guarantees persistence & tracks hits via `cachedContentTokenCount`.
4. Subagent MCP Stitching No orphaned traces	OTel API + MCP Transport	`propagation.inject(ctx, _meta)` & `propagation.extract(ctx, _meta)`	Keeps parent orchestrator and worker subagents in one unified tree.
5. Streaming OpenAI Usage Audit cached tokens	OpenAI SDK / Responses	`stream_options: { include_usage: true }` + read `cached_tokens`	Populates accurate cache-hit ledger rows even on SSE streams.

OpenTelemetry Collector Pipeline: Normalize & Route GenAI Telemetry

```
receivers:
  otlp: { protocols: { grpc: { endpoint: "0.0.0.0:4317" }, http: { endpoint: "0.0.0.0:4318" } } }
processors:
  transform/genai_semconv:
    trace_statements:
      - context: span
        statements:
          - set(attributes["gen_ai.provider.name"], attributes["gen_ai.system"]) where
attributes["gen_ai.provider.name"] == nil
  batch: { timeout: 2s, send_batch_size: 512 }
exporters:
  otlphttp/langfuse: { endpoint: "https://cloud.langfuse.com/api/public/otel" }
  otlp/arize_ax: { endpoint: "otlp.arize.com:443" }
```

Why Normalize Inside the OTel Collector?

Different agent frameworks emit slightly different `gen_ai.*` schema revisions. A five-line Collector `transform` processor lets you upgrade individual microservices independently while keeping Langfuse, Arize AX, and FinOps dashboards unified.

PII Redaction in Traces, Budget Alerts & FinOps Rollout

Production observability must protect customer secrets and stop runaway agent loops before they generate surprise invoices. Enforce privacy defaults at the Collector edge and roll out optimizations in measured stages.

1 Privacy-First Content Capture & Redaction

By OpenTelemetry default, message payloads (`gen_ai.input.messages` and `gen_ai.output.messages`) remain **off** so spans record only safe operational metadata: latency, model, finish reasons, and token counts.

When debugging requires prompt text in staging or sampled production runs, insert an OTel Collector `redaction` or `transform` regex filter to scrub API keys, bearer tokens, emails, and SSNs before OTLP export.

2 Automated Budget Alerts & Loop Breakers

Hard Runaway Caps: Enforce per-session ceilings on turn count (e.g., `max_turns = 20`), cumulative reasoning tokens, and USD spend directly inside the agent harness.

Anomaly Alerts: Trigger PagerDuty or Slack warnings whenever rolling 15-minute `cache_hit_ratio` drops below 70%, `finish_reasons` reports repeated `length` truncations, or cost per completed task spikes 2x above baseline.

Three-Phase Production FinOps Rollout Playbook

P1 Shadow Baseline

Enable `gen_ai.*` structural spans with content capture off. Measure baseline cost per task, p95 TTFT, and current accidental cache hit rate for 48 hours.

P2 Prefix Lock & Compact

Move timestamps and dynamic diffs out of system prompts, freeze tool arrays, add `cache_control` breakpoints, and enable batched tool compaction.

P3 Route & Eval Gate

Route classification and summarization steps to fast small models while validating online Langfuse or Arize AX correctness scores stay within 1% of baseline.

Compliance Warning: Never Bypass Sampling & Scrubbing in Regulated Workloads

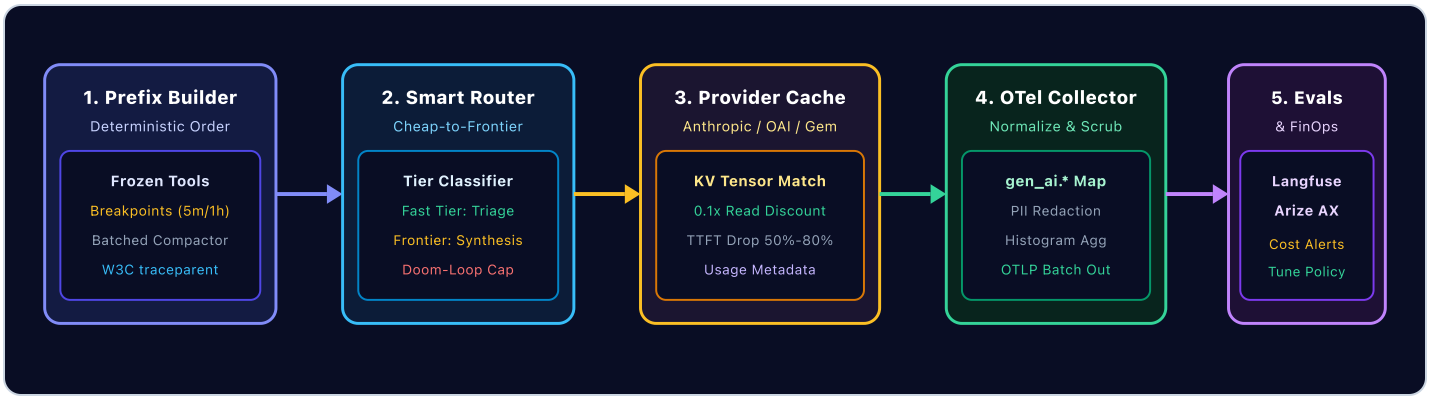
Enabling full prompt body capture in production without Collector-side PII redaction can replicate proprietary source code, credentials, and user data into external observability storage. Separate structural FinOps metrics (always 100% sampled) from redacted debug trace payloads (1% to 5% sampled or gated by error status).

Key FinOps Alert Thresholds & Automated Actions

Telemetry Signal	Alert Threshold	Automated Guardrail or Remediation
Rolling Cache Hit Ratio	`< 70%` over 15 min	Alert on prefix drift; diff recent system prompt or tool-schema commits.
Reasoning / Output Surge	`> 3x` task p95	Cap reasoning budget or escalate sub-task out of repetitive tool loop.
Length Truncation Rate	`finish_reasons: length` > 5%	Trigger server-side tool compaction earlier before context saturates.

End-to-End Synergy: Tracing, Caching, Routing & Evals in Harmony

How deterministic prompt builders, dynamic model routers, provider KV caches, OTel Collectors, and trace-linked evaluation platforms connect into one continuous cost-and-quality flywheel.



1 Closing the Eval-to-Routing Feedback Loop

Whenever an online LLM-as-a-Judge evaluator scores a dynamically routed fast-model turn below your quality threshold, your dashboard highlights the exact `trace_id` and `gen_ai.response.model`. Engineers can promote that specific task slice back to a frontier model or tune the compaction threshold with empirical evidence rather than guesswork.

2 Synergy Across Sandboxes, MCP & Coding Agents

Pairing isolated sandbox execution with OpenTelemetry tracing turns every bash run, test failure, and cache hit into one searchable timeline. When coding agents invoke tools over MCP or delegate to parallel subagents, `_meta.traceparent` links every sandbox span back to the user's original prompt and exact dollar cost.

Interoperability Standards Connecting the Stack

W W3C Trace Context

Standard `traceparent` and `tracestate` headers bind HTTP gateways, queues, and MCP calls into a single distributed trace.

O OpenTelemetry & OpenInference

Vendor-neutral `gen_ai.*` spans map cleanly into Langfuse, Arize AX, Datadog, and Prometheus backends without code rewrites.

L Deterministic Trace IDs

When async webhooks cannot forward OTel headers directly, seed matching trace IDs from a shared correlation token.

Unified Engineering Loop: Measure Before You Optimize

Teams that wire OTel GenAI tracing before tuning prompts save 2x to 4x faster: traces immediately pinpoint whether token spikes come from cache invalidation, unpruned tool output, or reasoning loops—replacing intuition with measurable cost-per-task engineering.

Troubleshooting Clinic & Pre-Flight Readiness Checklist

Rapid ten-second fixes for the seven most frequent caching and telemetry issues, followed by a six-point readiness gate before shipping autonomous agents to production.

Symptom & Telemetry Signal	Root Cause	10-Second Production Fix
`cache_read == 0` every turn	Dynamic timestamp, UUID, or reordered tool schema near top of prompt.	Move dynamic fields into the latest user turn; sort tool schemas deterministically.
`cache_creation` spikes every 6m	5-minute ephemeral TTL expires during human review or slow CI waits.	Add `"ttl": "1h"` on stable breakpoints (or OpenAI extended retention).
Gemini `400 Invalid Argument`	Explicit `CachedContent` payload below 2,048 / 4,096 minimum tokens.	Pad bundle past the model floor with shared repo docs or rely on implicit caching.
Orphaned subagent traces	Async worker queue or MCP call dropped W3C trace context headers.	Inject `traceparent` into MCP `_meta` and extract before child spans start.
Compaction triggers cost jump	Summarizing history on every turn busts the upstream KV cache prefix.	Run compaction only at high-water thresholds (e.g., 50K–64K tokens) in batches.
Missing TTFT or token split	Streaming omits final usage chunk or SDK emits legacy `gen_ai.system`.	Set `stream_options: { include_usage: true }` and map attributes in OTel Collector.
Secrets visible in trace logs	Opt-in message capture enabled without Collector scrubbing rules.	Disable `gen_ai.input.messages` in prod or enforce OTel `redaction` regex masks.

6-Point Pre-Flight Observability & Caching Checklist

☐ **1. Zero Volatile Prefix Data:** Confirm static tools, system instructions, and frozen history stay byte-identical across turns.

☐ **2. Cache Hit Ratio ≥ 75%:** Verify `gen_ai.usage.cache_read.input_tokens` dominates multi-turn input totals in staging.

☐ **3. Unified Subagent Trace Tree:** Confirm parent, tool, and MCP subagent spans share one `trace_id` with zero orphan roots.

☐ **4. PII Redaction Verified:** Confirm prompt message capture is off by default or scrubbed cleanly by the OTel Collector edge.

☐ **5. Compaction Threshold Tuned:** Ensure stale tool outputs are pruned in batches rather than busting cache every turn.

☐ **6. FinOps Caps & Evals Armed:** Enable turn/spend circuit breakers and trace-linked Langfuse or Arize AX regression alerts.