Día 17 · Tema 6 — Sistemas agénticos en producción: coste, latencia y observabilidad

Eduardo C. Garrido-Merchán · ecgarrido@comillas.edu Universidad Pontificia Comillas · ICADE

1. De prototipo a producción

Un agente que funciona en un notebook no funciona en producción. Los problemas adicionales son: **(i)** latencia bajo carga; **(ii)** coste a escala; **(iii)** fallos parciales (un tool externo cae); **(iv)** no-determinismo (mismo prompt → outputs distintos); **(v)** observabilidad para depurar.

2. Modelo de latencia de un agente

El tiempo total de una sesión de N turnos es

$$T_{\text{total}} = \sum_{t=1}^{N} \left(T_t^{\text{LLM}} + T_t^{\text{tool}} \right), \tag{1}$$

T_t^{LLM} tiempo de la llamada t al LLM (TTFT + decoding)
T_t^{tool} tiempo de ejecutar la(s) tool(s) llamada(s) en el paso t

T_t^{LLM} se descompone en *time-to-first-token* (latencia de red + prefill) más N_t^{out}/tps (decoding). Para Claude Sonnet 4.5 vía API, TTFT $\sim$ 0,7 s, tps $\sim$ 60 tok/s; una respuesta de 300 tokens cuesta $\sim$ 5,7 s antes de los tools.

3. Estrategias para bajar latencia

- **Prompt caching** (Día 11): el prefijo *caliente* se cobra al 30 % y se procesa al $\sim 5\times$ de velocidad.
- **Streaming**: enviar al usuario los tokens según se generan, reduciendo la latencia *percibida*.
- **Tools en paralelo**: si el agente decide invocar k tools independientes, lanzarlas concurrentemente.
- **Modelo más pequeño para tareas simples**: *router* que delega a Haiku 4.5 las queries triviales y reserva Sonnet 4.5 para las complejas.

4. Modelo de coste a escala

Para un servicio que recibe Q queries/día:

$$C_{\text{día}} = Q \cdot \left(\bar{C}_{\text{LLM}} + \bar{C}_{\text{tool}} \right), \quad \bar{C}_{\text{LLM}} = \frac{\bar{n}_{\text{in}}}{10^6} p_{\text{in}} + \frac{\bar{n}_{\text{out}}}{10^6} p_{\text{out}}. \tag{2}$$

Para $Q = 10\,000$, $\bar{n}_{\text{in}} = 8000$, $\bar{n}_{\text{out}} = 600$ y precios Sonnet, $\bar{C}_{\text{LLM}} = 0{,}033\,\$$ y $C_{\text{día}} = 330\,\$$. Anualizado: $\sim 120\,000\,\$$.

5. Trade-off coste-calidad

La calidad de respuesta tiende a saturar con el coste por consulta:

$$Q(\$) \approx Q_{\text{máx}} \left(1 - e^{-\$/\$_0} \right), \tag{3}$$

con $\$_0$ específico de la tarea. Empíricamente: pasar de Haiku (0,005 \$/q) a Sonnet (0,03 \$/q) sube la calidad significativamente; pasar de Sonnet a Opus (0,15 \$/q) sube poco salvo en razonamiento complejo. La elección por defecto en 2026 es Sonnet.

6. Observabilidad

Un agente productivo debe emitir, por sesión:

- `session_id`, `user_id`, `timestamp`.
- Traza completa: $\{(c_t, a_t, \text{exec}(a_t))\}_{t=1}^{N}$.
- Métricas: N, T_{total}, C, tasa de tool errors.
- Veredicto humano (opcional): pulgar arriba/abajo, comentario.

Herramientas estándar: Langfuse, Arize Phoenix, Weights & Biases Prompts, Helicone. **No usar nunca un agente sin observabilidad en producción**: depurar a oscuras es imposible.

7. Patrones anti-fallo

1. **Timeout en tools**: cada tool tiene un presupuesto temporal; si lo excede, retornar error y dejar al agente decidir.
2. **Retry con backoff**: errores transitorios (429, 503) se reintentan con espera exponencial; errores 4xx (lógicos) no.
3. **Circuit breaker**: si un tool externo falla repetidamente, no seguir llamándolo durante un periodo.
4. **Fallback path**: si Sonnet falla, intentar Haiku con warning; si Haiku también falla, devolver mensaje canónico.

8. Evaluación continua

Offline: dataset de *golden conversations* (50–500 sesiones ideales). Se ejecuta el agente sobre ellas y se mide la similitud de la respuesta con la gold (BLEU/ROUGE para texto, exact-match para acciones). *Online*: A/B test contra el agente anterior, medido en métricas de negocio (resolución, NPS, escalado).

Idea clave. Sin evaluación continua, cada cambio del prompt o del modelo es una apuesta. *El equipo que mide gana al equipo que improvisa.*