El mito de los tokens en arquitecturas de agentes

MCP vs Function Calling y la verdadera métrica del coste.



Basado en la investigación original de Cristian Palau.

Gemini Notebook

La Pregunta

¿Cuántos posts tengo sobre Docker?

El modelo no ahorra tokens porque le envías el menú entero en cada pedido


```
{
  "tools": [
      {
        "type": "function",
        "function": {
          "name": "contar_posts",
          "description": "Cuenta el número de posts publicados por
usuario sobre un tema especifico.",
          "parameters": {
            "type": "object",
            "properties": {
              "tema": {
                "type": "string",
                "description": "El tema sobre el que contar los p
e.g., 'Docker', 'Python'.",
              "usuario_id": {
                "type": "string",
                "description": "El ID único del usuario."
              }
            },
            "required": ["tema", "usuario_id"]
          }
        }
      },
      {
        "type": "function",
        "function": {
          "name": "listar_comentarios",
          "description": "Lista los últimos comentarios del usuario.",
          "parameters": {
            "type": "object",
            "properties": {
              "tema": {
              "type": "string",
              "description": "El tema sobre el que contar los posts, e.
"",
            },
            "type": "object",
            "properties": {
              "tema": {
                "type": "string",
              "usuario_id": {
              "type": "string",
              "description": "Python."
              }
            },
            "usuario_id": {
              "type": "string",
              "description": "El ID único del usuario."
            }
          }
        },
        {
```


Payload oculto: 200 definiciones de herramientas completas enviadas en CADA petición.

Gemini Notebook

Separación de responsabilidades: El Camarero y La Cocina

El Modelo
(El Camarero)



- Lee y escribe texto.
- Decide qué herramienta usar.
- NO tiene acceso a bases de datos.
- No toca la infraestructura.

El modelo decide, el agente actúa.

El Agente
(La Cocina)



- El código que tú escribes.
- Ejecuta las funciones reales.
- Posee los permisos y credenciales.

Gemini Notebook

Anatomía del Function Calling: El catálogo viaja dos veces



Gemini Notebook

Function Calling vs MCP: Monolito frente a Ecosistema Distribuido

Function Calling



Un monolito. Todo el código y las descripciones viven en un solo lugar.

MCP (Model Context Protocol)



Un **ecosistema distribuido**. Estandariza la conexión usando JSON-RPC 2.0. Múltiples clientes consultan el mismo catálogo.

MCP no cambia cómo el modelo pide la herramienta, cambia de dónde saca el agente la descripción.

Gemini Notebook

Matriz de Diagnóstico Arquitectónico

Dimensión	Function Calling	MCP
¿Qué es?	Formato de petición Habilidad entrenada	Protocolo de conexión cliente-servidor
¿Quién describe las herramientas?	Tú, en cada petición al LLM.	El servidor, cuando el agente se conecta.
¿Dónde residen las herramientas?	En el código de tu agente local.	En el servidor (local o remoto).
¿Quién ejecuta la acción?	Tu agente.	El servidor (con sus propios permisos).
¿Qué problema principal resuelve?	Obtener respuestas estructuradas (JSON).	Integrar N aplicaciones con M herramientas sin repetir código.

Conclusión: Ninguno resuelve el formato de la petición final al LLM. MCP se apoya obligatoriamente en Function Calling.

Gemini Notebook

El Banco de Pruebas: Aislamiento de variables

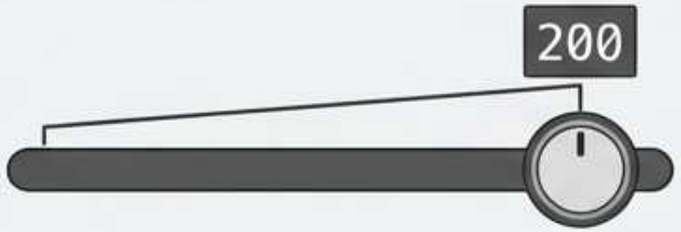
Constantes (Variables de Control)

Modelo: "deepseek-chat"

Pregunta: "¿Cuántos posts tengo sobre Docker?"

Herramienta Objetivo: contar_posts(tema)

Variable Independiente (Ruido)



Inclusión de 1 a 200 distractores. Herramientas falsas (ej. contar_visitas) inyectadas para confundir al modelo y saturar el contexto.

Las 4 Vías Probadas

1. Function Calling clásico
2. MCP (vía stdio local)
3. MCP (vía HTTP remoto)
4. Descubrimiento Progresivo

Gemini Notebook

El Veredicto: El protocolo no reduce el payload



Function Calling y MCP gastan EXACTAMENTE los mismos tokens.

La descripción completa de las 200 herramientas viaja íntegra al LLM en ambos casos.

Gemini Notebook

La verdadera solución: Descubrimiento Progresivo

Paso 1: El Buscador



El Modelo no recibe 200 herramientas. Recibe **SOLO UNA** herramienta: `buscar_tools`.

Paso 2: La Consulta



El Modelo pide ayuda: “Busca herramientas para contar posts”.

Paso 3: El Índice



El Agente busca localmente en su catálogo. Descarta 197 distractores.

Paso 4: La Entrega



El Modelo recibe ÚNICAMENTE las 3 descripciones relevantes y elige la correcta.

El catálogo **masivo** nunca viaja.

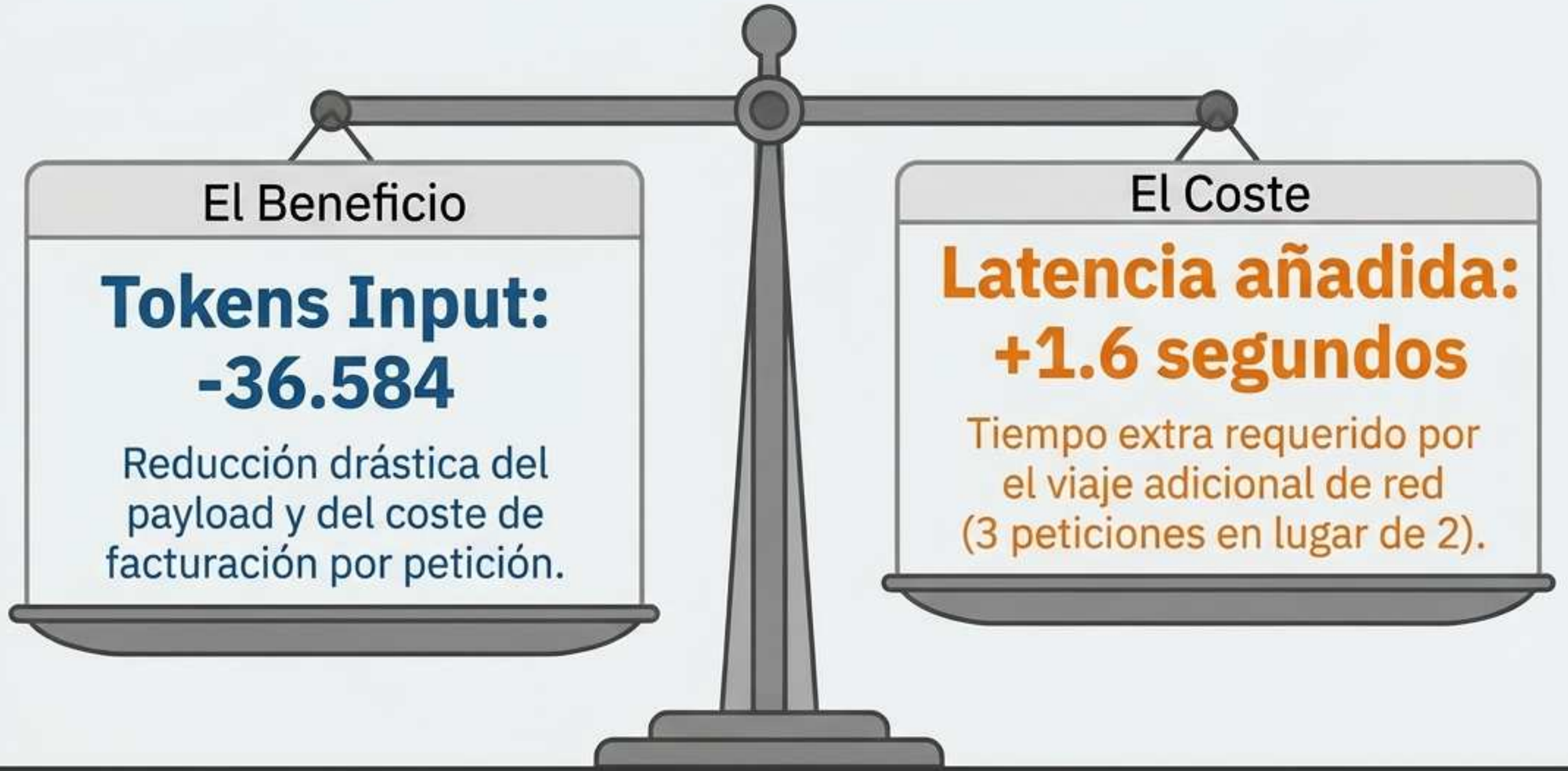
Gemini Notebook

El colapso del coste: 18 veces más barato



Gemini Notebook

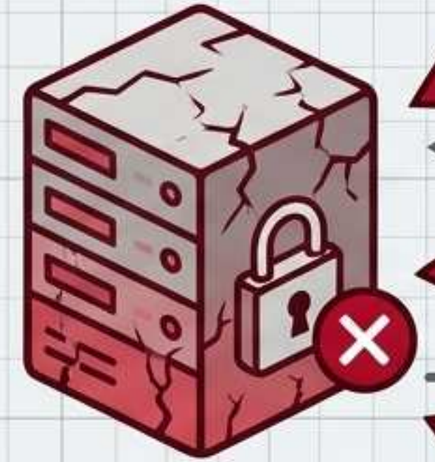
El Trade-Off arquitectónico: Tokens frente a Latencia



En la práctica: En sistemas donde la generación de la respuesta del LLM es el verdadero cuello de botella, este segundo y medio extra es insignificante frente al brutal ahorro económico.

Gemini Notebook

MCP y la falsa sensación de seguridad



Servidor MCP

Tool Poisoning

Inyección de instrucciones maliciosas en la descripción de la herramienta. El LLM las obedece como si fueran su System Prompt.

El Cambiazo (*Rug Pull*)

El servidor muestra una herramienta benigna para obtener tu aprobación manual, pero altera el código real justo en el momento de la ejecución.

Tool Shadowing

Un servidor malicioso secuestra o sobrescribe las descripciones de herramientas legítimas para desviar datos (ej. copiar correos a otra dirección).

Gemini Notebook

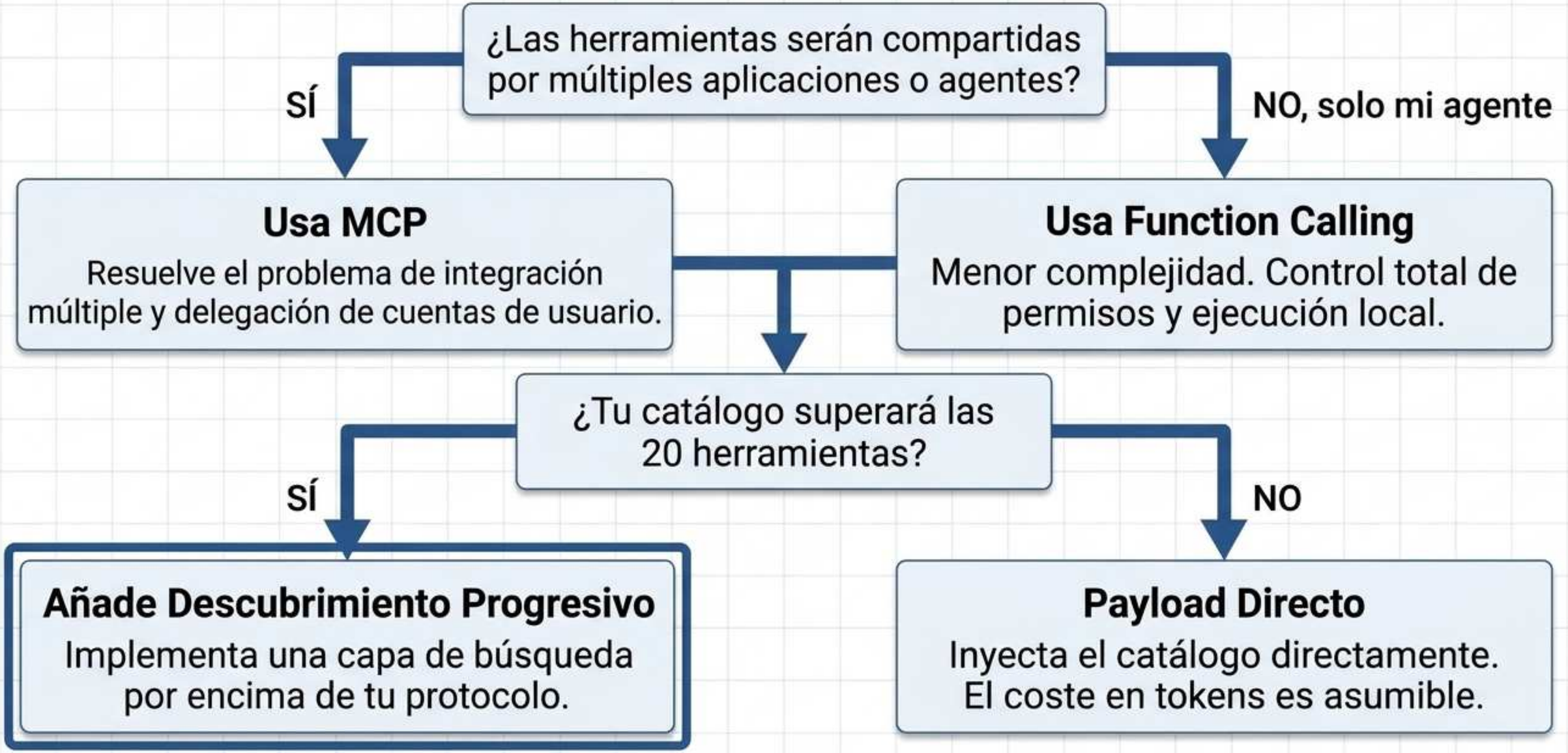
Framework de Mitigación para MCP

Concepto	Objetivo	Acción
 Integridad Criptográfica	 Defensa contra Rug Pull	Guardar hashes de la definición de la herramienta al aprobarla. Rechazar la ejecución si el hash del servidor cambia remotamente.
 Aislamiento OAuth	 Defensa contra Confused Deputy	Exigir que el servidor MCP ejecute las acciones con los permisos del usuario que realiza la petición, NUNCA con permisos globales de servicio.
 Host Enforcement	 Defensa contra Permisos Excesivos	El Host (tu agente) debe mostrar siempre el payload completo antes de la aprobación humana, no solo el nombre.

Tú no controlas el servidor, pero controlas el Host.

Gemini Notebook

Síntesis Arquitectónica: Árbol de Decisión



Gemini Notebook

Principios Fundamentales

// 1. El protocolo no salva el contexto, el tamaño del catálogo sí.

Ni MCP ni Function Calling reducen el payload mágico. La descripción viaja igual.

// 2. MCP es para escalabilidad de integración, no para rendimiento.

Úsalo exclusivamente para evitar escribir código duplicado entre Cursor, Claude y tu App.

// 3. Indexa tus herramientas si cruzas la barrera de las 20.

El Descubrimiento Progresivo cambia una latencia aceptable por un ahorro del 94% en la factura de la API.

Investigación original y benchmark de tokens por Cristian Palau. Deck adaptado para diseño arquitectónico de sistema

Gemini Notebook