Code Mode vs Traditional MCP

A practical, example-driven reference for understanding execution models, performance, and hybrid LLM-in-the-loop architectures.

1. Mental Models (At a Glance)

Traditional MCP: LLM orchestrates tools step-by-step inside the chat loop.
Code Mode: LLM writes an executable program and hands control to a runtime.

```
Traditional MCP: User → LLM → Tool → LLM → Tool → LLM → Answer
```

```
Code Mode: User → LLM → Program → Runtime → Answer
```

2. Example: Traditional MCP (Tool-Oriented)

Use case: Analyze logs and send an alert if there is an anomaly.

```
1. LLM requests logs via tool call 2. LLM waits for response 3. LLM analyzes logs 4. LLM
calls alert tool 5. LLM responds to user
```

- Multiple blocking round-trips
- Intermediate state stored in tokens
- Higher latency and cost

3. Example: Code Mode (Executable Plan)

The LLM reasons once, then emits a complete executable program.

```
for each log_batch: update error_count if error_count > threshold: send alert exit
```

- Single execution handoff
- Real variables and loops
- Early exit and streaming possible

4. Hybrid Architecture (LLM-in-the-loop + Code Mode)

```
■■■■■■■■■■■■■ ■ User ■ ■■■■■■■■■■■■■ ■ ▼ ■■■■■■■■■■■■■■■■■■■■ ■ LLM Planner ■
(reasoning, semantics) ■■■■■■■■■■■■■■■■■■■■ ■ emits program ▼
■■■■■■■■■■■■■■■■■■■■ ■ Code Runtime ■ (state, loops, execution)
■■■■■■■■■■■■■■■■■■■■ ■ needs judgment? ▼ ■■■■■■■■■■■■■■■■■■■■ ■ LLM Evaluator ■
(semantic decision) ■■■■■■■■■■■■■■■■■■■■ ■ ▼ ■■■■■■■■■■■■■ ■ Response ■
■■■■■■■■■■■■■
```

5. Example: Hybrid Execution

Most real systems combine deterministic execution with selective LLM judgment.

```
if metrics.look_weird: judgment = LLM.evaluate(context) if judgment.is_real_issue:
alert(judgment.reason)
```

- Code controls flow and performance
- LLM used only for semantic judgment
- Predictable, fast, and auditable

6. What Belongs Where

Put in Code: thresholds, retries, loops, aggregation, state.
Put in LLM: ambiguity, intent, quality judgment, explanation.

Key takeaway: LLMs decide what logic should exist; code decides when and how fast it runs.