Stop Using Frontier AI for Everything

February 4, 2026

Download this post as PDF

📚 **More on this topic:** Ollama Troubleshooting Guide · llama.cpp vs Ollama vs vLLM · VRAM Requirements · Budget AI PC Under $500 · Planning Tool

If you're using Claude Opus or GPT-4 to check file syntax, format JSON, or answer "what's the capital of France?" — you're burning money.

Frontier models cost 60x more than their smaller siblings. Most tasks don't need that power. This guide shows you how to build a tiered AI strategy that uses the right model for each task, saving hundreds of dollars a month without sacrificing quality where it matters.

The Problem: Frontier Models for Everything

Here's what happens when you use Opus/GPT-4 for everything:

Simple question: "What's the syntax for a Python list comprehension?"

- Haiku cost: $0.0003
- Opus cost: $0.02
- **Overpay:** 67x

Format this JSON:

- Local model cost: $0
- Opus cost: $0.01
- **Overpay:** ∞

Analyze this architecture and identify security vulnerabilities:

- Haiku: Misses subtle issues
- Opus: Catches everything
- **Right tool for the job**

The pattern is clear: frontier models are overkill for 80% of tasks, but essential for the remaining 20%.

The Tiered Model Hierarchy

Tier	Model Examples	Cost (per 1M tokens)	Best For
Tier 1: Local	Llama 3.1 8B, Qwen 2.5 7B, Mistral 7B	$0 (electricity only)	File operations, formatting, basic lookups
Tier 2: Fast/ Cheap	Claude Haiku, GPT-4o mini	$0.25-0.60 output	Simple reasoning, classification, summaries
Tier 3: Capable	Claude Sonnet, GPT-4o	$3-15 output	Complex coding, analysis, most work tasks
Tier 4: Frontier	Claude Opus, GPT-4, o1	$15-60 output	Architecture, security, critical decisions

Cost Comparison

Running 1,000 tasks (average 2K tokens each):

Strategy	Cost
All Opus	$30.00
All Sonnet	$6.00
Tiered (smart routing)	$2.50
All local	$0.15 (electricity)

Tiered approach saves 88% vs all-Opus while maintaining quality where needed.

Tier 1: Local Models (Free)

Local models running on your own hardware cost nothing per query. The only cost is electricity (~$0.01/hour for an RTX 3060).

What Local Models Handle Well

Task	Example	Why Local Works
File syntax checks	“Is this JSON valid?”	Deterministic, no reasoning needed

Task	Example	Why Local Works
Formatting	“Convert this to markdown table”	Pattern matching, no creativity
Basic lookups	“What port does SSH use?”	Factual recall, no nuance
Code completion	Simple autocomplete	Pattern continuation
Text transformation	“Make this lowercase”	Direct manipulation
Template filling	“Fill this form with these values”	Mechanical substitution

Recommended Local Models

Model	VRAM	Speed	Best For
Llama 3.1 8B Q4	5GB	~40 tok/s	General tasks
Qwen 2.5 7B Q4	5GB	~45 tok/s	Coding, multilingual
Mistral 7B Q4	4.5GB	~45 tok/s	Fast responses
DeepSeek Coder 6.7B	4.5GB	~40 tok/s	Code-specific

Setup: Install Ollama and run `ollama pull llama3.1:8b`. Done.

Local Model Limitations

Don’t use local for:

- Complex reasoning chains
- Nuanced analysis
- Security-critical decisions
- Tasks requiring broad knowledge
- Anything where being wrong has consequences

Tier 2: Fast/Cheap Cloud (Haiku, GPT-4o mini)

When local isn’t confident enough but you don’t need heavy reasoning.

Haiku/Mini Sweet Spot

Task	Example	Why This Tier
Classification	“Is this email spam or not?”	Binary decision, clear criteria
Simple summaries	“Summarize this paragraph”	Compression, not analysis
Basic Q&A	“Explain what a mutex does”	Factual with light reasoning
Code review (simple)	“Any obvious bugs here?”	Pattern recognition
Data extraction	“Pull the dates from this text”	Structured parsing
Light reasoning	“Which option is cheaper?”	Simple comparison

Cost at This Tier

Model	Input	Output	1000 Simple Tasks
Claude 3.5 Haiku	$0.25/M	$1.25/M	~$1.50
GPT-4o mini	$0.15/M	$0.60/M	~$0.75

When to escalate up: Model says “I’m not sure” or gives clearly wrong answers.

Tier 3: Capable Models (Sonnet, GPT-4o)

Your workhorse tier. Handles 80% of real work tasks.

Sonnet/GPT-4o Sweet Spot

Task	Example	Why This Tier
Complex coding	“Refactor this function with better error handling”	Requires understanding intent
Analysis	“What are the tradeoffs of these two approaches?”	Nuanced comparison
Writing	“Write documentation for this API”	Coherent, structured output
Debugging	“Why is this test failing?”	Multi-step reasoning
	“Review this PR for correctness and style”	Pattern + logic

Task	Example	Why This Tier
Code review (thorough)		
Research synthesis	“Summarize these 5 papers”	Integration across sources

Cost at This Tier

Model	Input	Output	1000 Medium Tasks
Claude 3.5 Sonnet	$3.00/M	$15.00/M	~$18.00
GPT-4o	$2.50/M	$10.00/M	~$12.50

When to escalate up: High-stakes decisions, security, architecture, legal.

Tier 4: Frontier Models (Opus, GPT-4, o1)

Reserve for tasks where being wrong is expensive.

When Frontier Is Worth It

Task	Why Frontier
Security review	Missing a vulnerability costs more than the API call
Architecture decisions	Wrong foundation = expensive rework
Legal/compliance	Mistakes have regulatory consequences
Complex debugging	Subtle bugs need deep reasoning
Critical code paths	Payment processing, auth, data handling
Novel problems	No clear pattern to follow

Cost at This Tier

Model	Input	Output	10 Critical Tasks
Claude Opus 4	$15.00/M	$75.00/M	~$1.80
GPT-4	$30.00/M	$60.00/M	~$1.80

Model	Input	Output	10 Critical Tasks
o1	$15.00/M	$60.00/M	~$1.50

The math: Even at 60x the cost of Haiku, 10 critical tasks per day = $1.80/day = $54/month. Cheap insurance.

The Escalation Pattern

Start cheap. Escalate only when needed.

```
Task arrives
    ↓
Can local handle it? (formatting, syntax, basic lookup)
    → Yes: Use local (free)
    → No: Continue
    ↓
Is it simple reasoning? (classification, basic Q&A)
    → Yes: Use Haiku/mini (~$0.001/task)
    → No: Continue
    ↓
Is it standard work? (coding, analysis, writing)
    → Yes: Use Sonnet/GPT-4o (~$0.02/task)
    → No: Continue
    ↓
Is it critical? (security, architecture, high-stakes)
    → Yes: Use Opus/GPT-4 (~$0.20/task)
```

Automatic Escalation Signals

Escalate when the current tier:

- Says “I’m not confident”
- Gives an obviously wrong answer
- Produces inconsistent output
- Can’t follow complex instructions
- Misses context from earlier in conversation

Practical Routing Examples

Example 1: Code Review Pipeline

Step	Task	Model	Cost
1	Check syntax/linting	Local (Llama 3.1 8B)	$0
2	Basic style violations	Haiku	$0.001
3	Logic review, edge cases	Sonnet	$0.02
4	Security review (if auth/payment)	Opus	$0.20

Total for typical PR: $0.02-0.22 (vs $0.80+ if all Opus)

Example 2: Documentation Generation

Step	Task	Model	Cost
1	Extract function signatures	Local	$0
2	Generate basic docstrings	Haiku	$0.001
3	Write usage examples	Sonnet	$0.02
4	Review for accuracy	Sonnet	$0.02

Total: ~$0.04 per function documented

Example 3: Bug Investigation

Step	Task	Model	Cost
1	Parse error logs	Local	$0
2	Identify likely cause	Sonnet	$0.02
3	Suggest fix	Sonnet	$0.02
4	Verify fix doesn’t introduce issues	Opus (if critical path)	$0.20

Total: $0.04-0.24 depending on criticality

Example 4: AI Coding Agent (Claude Code, Cursor, etc.)

Most AI coding tools let you choose models. Smart routing:

Task Type	Model	Why
File reads, grep, ls	Local or skip LLM	No reasoning needed
Simple edits	Haiku	Pattern application
New feature implementation	Sonnet	Balance of speed and quality
Architecture changes	Opus	Get it right the first time
Security-sensitive code	Opus	Always

When to Always Use Frontier

Some tasks should never be cheap-routed:

Always Opus/GPT-4

Category	Examples
Security	Auth flows, encryption, input validation, API security
Financial	Payment processing, billing logic, financial calculations
Legal	Compliance checks, terms generation, regulatory code
Architecture	System design, database schema, API contracts
Data integrity	Migration scripts, data transformation, backup logic
Novel problems	First time solving this type of problem

The 10x Rule

If fixing a mistake costs 10x more than the API call difference, use frontier.

- Opus costs $0.20 more than Sonnet for a task
- Fixing a security bug in production costs $2,000 (incident response, patching, customer communication)
- **Use Opus.**

Setting Up a Tiered Workflow

For Individual Developers

1. **Install Ollama** for local models: `curl -fsSL https://ollama.com/install.sh | sh`
2. **Pull a local model:** `ollama pull llama3.1:8b`
3. **Use local for:** file checks, formatting, basic questions
4. **Use Sonnet/GPT-4o for:** daily coding work
5. **Use Opus for:** anything security-related or architectural

For API Users

Route requests programmatically:

```
def select_model(task_type, is_critical=False):
    if is_critical:
        return "claude-opus-4"

    simple_tasks = ["format", "syntax", "lookup", "classify"]
    medium_tasks = ["code", "analyze", "write", "debug"]

    if task_type in simple_tasks:
        return "claude-3-5-haiku-latest"
    elif task_type in medium_tasks:
        return "claude-sonnet-4-20250514"
    else:
        return "claude-opus-4"
```

For AI Agents

Configure your agent to use tiered models:

```
# Example agent config
models:
  file_operations: ollama/llama3.1:8b
  simple_reasoning: claude-3-5-haiku-latest
  coding: claude-sonnet-4-20250514
  critical: claude-opus-4
```

```
routing:
  - pattern: "read file|list directory|check syntax"
    model: file_operations
  - pattern: "is this|classify|simple question"
    model: simple_reasoning
  - pattern: "security|auth|payment|architecture"
    model: critical
  - default: coding
```

Cost Savings Calculator

Before (All Frontier)

Usage	Tasks/Day	Tokens/Task	Model	Daily Cost	Monthly
Light	50	2K	Opus	$7.50	$225
Medium	200	2K	Opus	$30.00	$900
Heavy	500	2K	Opus	$75.00	$2,250

After (Tiered)

Usage	Tier Mix	Daily Cost	Monthly	Savings
Light	60% local, 30% Haiku, 10% Sonnet	$0.75	$22.50	90%
Medium	50% local, 30% Haiku, 15% Sonnet, 5% Opus	$4.50	$135	85%
Heavy	40% local, 35% Haiku, 20% Sonnet, 5% Opus	$12.50	$375	83%

Common Mistakes

Mistake 1: Routing Everything to Local

Local models are free but limited. Using them for complex reasoning leads to:

- Wrong answers that waste your time

- Bugs that cost more to fix than the API call
- Frustration with “AI doesn’t work”

Fix: Use local for mechanical tasks only.

Mistake 2: Fear of Frontier Costs

Avoiding Opus for security reviews to save $0.20 is false economy when a missed vulnerability costs thousands.

Fix: Always use frontier for critical tasks.

Mistake 3: No Escalation Path

Using Haiku and accepting bad answers instead of escalating.

Fix: When a model says “I’m not sure,” escalate immediately.

Mistake 4: Context Bloat

Sending 50K tokens of context to Opus when Sonnet with 10K tokens would suffice.

Fix: Trim context aggressively. Smaller context = faster + cheaper.

The Bottom Line

The tiered model strategy:

1. **Local (free):** File operations, formatting, basic lookups
2. **Haiku ($0.25/M):** Simple reasoning, classification
3. **Sonnet ($3/M):** Daily coding work, analysis
4. **Opus ($15/M):** Security, architecture, critical decisions

The rules:

- Start at the cheapest tier that might work
- Escalate when confidence drops
- Always use frontier for high-stakes tasks
- Don’t penny-pinch on security

The math:

- Tiered routing saves 80-90% vs all-frontier
- Even heavy users spend <$400/month with smart routing
- The savings pay for a dedicated local AI setup in 2 months

Build the habit of asking "what tier does this task need?" before every AI interaction. Your API bill will thank you.

Related Guides

- Ollama Troubleshooting Guide
- llama.cpp vs Ollama vs vLLM
- VRAM Requirements for Local LLMs
- Budget AI PC Under $500
- How Much Does It Cost to Run LLMs Locally
- Local LLMs vs ChatGPT
- Local AI Planning Tool — VRAM Calculator

Source: https://insiderllm.com/blog/tiered-ai-model-strategy/