OpenClaw Model Routing: Cheap Models for Simple Tasks, Smart Models When Needed

February 14, 2026

Download this guide as PDF

> **Quick Answer:** Add a fallbacks array and heartbeat model to your openclaw.json. Set Haiku as your primary (it handles 75% of agent work), Ollama for heartbeats (free), and Opus in the fallback chain for when Haiku chokes. A 6-hour overnight task with 14 sub-agents cost $6 with this setup. The same task on Sonnet alone would have been $150. ClawRouter automates this further with 15-dimension task scoring that routes to the cheapest capable model in under 1ms.

📚 **More on this topic:** OpenClaw Token Optimization · Best Models for OpenClaw · Tiered Model Strategy · OpenClaw 100% Local · OpenClaw Setup Guide · Planning Tool

OpenClaw's default config points every request at one model. Every heartbeat, every file rename, every "is this JSON valid?" goes to the same place. If that place is Claude Opus, you're paying $15 per million input tokens for work that a free local model handles identically.

One user loaded $25 onto Anthropic and watched it drain to $5 in a day with the agent doing nothing. Heartbeats were pinging Opus every 30 minutes, loading full context each time. That's roughly $2-5/day in idle costs before you ask the agent to do a single useful thing.

Model routing fixes this. You tell OpenClaw which model handles which type of work, and the agent routes accordingly. The hard tasks go to Opus. Everything else goes to something cheaper. Your agent doesn't get dumber. Your bill gets smaller.

The One-Model Problem

Here's what running everything through Opus looks like:

Activity	Frequency	Tokens Per Call	Daily Cost (Opus)
Heartbeats	48/day (every 30 min)	50-100K each	$3-5
File operations	10-20/day	5-10K each	$0.50-1
Simple lookups	5-10/day	2-5K each	$0.25-0.50
Actual reasoning tasks	3-5/day	10-50K each	$1-3

Activity	Frequency	Tokens Per Call	Daily Cost (Opus)
Daily total			**$5-10**

That's $150-300/month. And 60-80% of it is overhead that never needed a frontier model.

The math is harsh. A heartbeat loads your context files, checks task state, and reports "system OK." That's it. It doesn't need the model that solved the restaurant phone call problem. It needs the model equivalent of checking your pulse.

How OpenClaw's Model Routing Works

OpenClaw's config supports multiple models at the provider level and routes different types of work to different models. The routing happens through three mechanisms:

1. Primary + Fallback Chain

Every agent has a primary model and an ordered list of fallbacks. If the primary fails (rate limit, outage, auth error), OpenClaw walks down the fallback chain until something works.

```
Request arrives → Try primary model
    → Success: Done
    → Failure: Try fallback[0]
        → Failure: Try fallback[1]
            → Failure: Try fallback[2]
                → All failed: Report error
```

Cooldown between retries follows a backoff pattern: 1 minute, then 5 minutes, then 25 minutes, capping at 1 hour. This prevents hammering a rate-limited API.

2. Heartbeat Routing

Heartbeats get their own model field, separate from the primary. This is the biggest routing decision you can make. Pointing heartbeats at a free local model eliminates idle costs.

3. Sub-Agent Routing

When your agent spawns sub-agents for parallel tasks, those sub-agents can use a different model than the parent. Research scouts get Haiku, the final report writer gets Sonnet, and the file organizer gets Ollama.

Which Tasks Need Which Model

Not all agent work is equal. Here's a concrete breakdown:

Free Tier: Local (Ollama)

Task	Why Local Works	Tokens Saved vs Opus
Heartbeats (48/day)	Status check, no reasoning	2-5M tokens/day
File moves and renames	Shell commands, no intelligence needed	50-100K/day
CSV compilation	Data formatting, mechanical work	100-500K/day
Log parsing	Pattern matching, structured input	50-200K/day
Data cleanup	Removing duplicates, fixing headers	100-300K/day

Cost: $0 (electricity only)

Cheap Tier: Haiku ($1/M input, $5/M output)

Task	Why Haiku Works	When to Escalate
Web research	Reading and extracting, not reasoning	If synthesis across 10+ sources needed
Email triage	Classification task	If crafting a nuanced reply
Data extraction	Structured parsing	If ambiguous data formats
Basic Q&A	Factual recall	If answer requires inference
Simple code tasks	Pattern application	If debugging a subtle bug

Cost: ~$0.006 per typical request

Capable Tier: Sonnet ($3/M input, $15/M output)

Task	Why Sonnet	When to Escalate
Cold outreach writing	Needs personality and persuasion	If writing for C-suite audiences
Code generation	Complex logic, multiple files	If architecting a new system
Analysis and reports	Multi-source synthesis	If the analysis drives a major decision
Debugging	Multi-step reasoning	If the bug is in security-critical code

Task	Why Sonnet	When to Escalate
Skill creation	Building new agent capabilities	If the skill is complex or safety-relevant

Cost: ~$0.03-0.05 per typical request

Frontier Tier: Opus ($15/M input, $75/M output)

Task	Why Only Opus
Architecture decisions	Wrong foundation = expensive rework
Security-sensitive operations	Missing a vulnerability costs more than the API call
Novel problem-solving	No established pattern to follow
Long multi-step chains (10+)	Cheaper models lose track of state
Critical business logic	Getting it wrong has real consequences

Cost: ~$0.20-1.00 per typical request

The rule: If fixing a mistake from a cheaper model costs more than the difference between Haiku and Opus, use Opus.

Setting Up Tiered Routing in openclaw.json

Starter Config: Two Tiers (Local + One API)

This is the minimum viable routing setup. Heartbeats go local, everything else goes to your API model of choice.

```
{
  "models": {
    "providers": {
      "ollama": {
        "baseUrl": "http://127.0.0.1:11434/v1",
        "apiKey": "ollama-local",
        "api": "openai-completions",
        "models": [
          { "id": "llama3.1:8b" }
        ]
      },
```

```
      "anthropic": {
        "apiKey": "sk-ant-your-key-here",
        "models": [
          { "id": "claude-sonnet-4-20250514" }
        ]
      }
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "anthropic/claude-sonnet-4-20250514"
      },
      "heartbeat": {
        "every": "30m",
        "model": "ollama/llama3.1:8b"
      }
    }
  }
}
```

What this saves: All idle costs eliminated ($2-5/day), but active tasks still run at Sonnet pricing.

Full Four-Tier Configuration

This is the setup that turned a $150 overnight task into $6.

```
{
  "models": {
    "providers": {
      "ollama": {
        "baseUrl": "http://127.0.0.1:11434/v1",
        "apiKey": "ollama-local",
        "api": "openai-completions",
        "models": [
          { "id": "llama3.1:8b" }
        ]
      },
      "anthropic": {
        "apiKey": "sk-ant-your-key-here",
        "models": [
          { "id": "claude-3-5-haiku-latest" },
          { "id": "claude-sonnet-4-20250514" },
          { "id": "claude-opus-4" }
        ]
```

```
      }
    },
    "aliases": {
      "haiku": "anthropic/claude-3-5-haiku-latest",
      "sonnet": "anthropic/claude-sonnet-4-20250514",
      "opus": "anthropic/claude-opus-4"
    }
  },
  "agents": {
    "defaults": {
      "model": {
        "primary": "anthropic/claude-3-5-haiku-latest",
        "fallbacks": [
          "anthropic/claude-sonnet-4-20250514",
          "anthropic/claude-opus-4"
        ]
      },
      "heartbeat": {
        "every": "30m",
        "model": "ollama/llama3.1:8b"
      },
      "subagents": {
        "model": "anthropic/claude-3-5-haiku-latest",
        "maxConcurrent": 3
      }
    }
  }
}
```

Notice three things:

1. **Primary is Haiku, not Sonnet.** This is deliberate. Haiku handles 75% of agent work competently. Making it the default means most requests go cheap.
2. **Fallbacks escalate upward.** Haiku → Sonnet → Opus. When Haiku can't handle something (or gets rate-limited), OpenClaw automatically moves up.
3. **Sub-agents default to Haiku.** When your agent spawns research scouts, they don't need Opus. Haiku reads blogs and extracts data just fine.

Model Aliases

The `aliases` block lets you reference models by short names instead of full provider/model strings. Useful when switching models via the `/model` command during a session:

```
/model opus    ← switches to Claude Opus
/model haiku   ← switches back to Haiku
```

Skill-Level Routing

OpenClaw skills can spawn sub-agents with specific model overrides. A coding skill might route to Sonnet while a file-organization skill routes to Ollama:

```
Coding skill → spawns sub-agent with model: "anthropic/claude-sonnet-4-20250514"
Research skill → spawns sub-agent with model: "anthropic/claude-3-5-haiku-latest"
File ops skill → spawns sub-agent with model: "ollama/llama3.1:8b"
```

You configure this in each skill's definition, not in the global config. The parent agent's model handles orchestration; the sub-agent's model handles execution.

ClawRouter: Automatic Task Classification

Setting up manual routing works, but you're still choosing which model handles each task. ClawRouter automates that decision.

What It Does

ClawRouter is an open-source routing layer by BlockRun AI that sits between OpenClaw and your model providers. It analyzes each request using a 15-dimension weighted scoring system and routes to the cheapest model that can handle the task. The classification happens locally in under 1 millisecond with zero API calls.

The Four Routing Tiers

Tier	Task Type	Model Examples	Approx. Cost
SIMPLE	File ops, formatting, status checks	Free/ultra-cheap models	~$0.001/M tokens
MEDIUM	Research, extraction, basic coding	Mid-tier models	~$1.50/M tokens
COMPLEX	Analysis, complex coding, writing	Gemini 2.5 Pro, Sonnet	~$10/M tokens
REASONING	Architecture, debugging, novel problems	Grok, Opus	~$0.50-15/M tokens

Routing Profiles

ClawRouter ships with four profiles:

- **auto** — Balanced routing (default). Good starting point.
- **eco** — Maximizes savings (78-99% cost reduction). Routes aggressively to cheap models.
- **premium** — Routes to the best model for each task type. Still cheaper than single-model.
- **free** — Uses only free models. Falls back to gpt-oss-120b when wallet is empty.

Installation

```
curl -fsSL https://blockrun.ai/ClawRouter-update | bash
openclaw gateway restart
```

ClawRouter uses USDC micropayments on Base for pay-per-request billing. $5 in your wallet covers thousands of requests at typical usage. When the balance hits zero, it falls back to free models automatically.

Is It Worth It?

For users spending $50+/month on API costs, yes. ClawRouter's blended average across all tiers is roughly $2/M tokens compared to $25/M for Opus. That's a 92% reduction with no manual routing decisions.

For users already running the four-tier config above, the incremental savings are smaller. ClawRouter's main value is convenience: it classifies tasks automatically instead of you defining routing rules.

Fallback Configuration

What Happens When a Model Fails

Three common failure modes and how OpenClaw handles each:

Failure	What OpenClaw Does	Your Config Controls
Rate limit (429)	Tries next model in fallback chain	`fallbacks` array order

Failure	What OpenClaw Does	Your Config Controls
Auth error	Rotates auth profiles within same provider first, then falls back	Provider auth config
Model outage	Moves to next fallback after cooldown	Cooldown is automatic (1min → 5min → 25min → 1hr cap)

Setting Up an Escalation Chain

The `fallbacks` array is ordered. Put cheaper models first:

```
"model": {
  "primary": "anthropic/claude-3-5-haiku-latest",
  "fallbacks": [
    "anthropic/claude-sonnet-4-20250514",
    "openai/gpt-4o",
    "anthropic/claude-opus-4"
  ]
}
```

This means: try Haiku first. If rate-limited, try Sonnet. If Sonnet is also down, try GPT-4o (different provider, different rate limits). Last resort: Opus.

Cross-provider fallbacks matter. When Anthropic hits a rate limit, OpenAI probably hasn't. Having providers from both means your agent rarely stalls completely.

Rate Limit Pacing for New Accounts

New Anthropic accounts get roughly 30,000 tokens per minute. That's tight. A single request with bloated context can blow through it.

Add pacing to your agent's operating instructions:

> Space API calls at least 5 seconds apart. If you receive a 429 error, wait 60 seconds before retrying. Do not fire multiple parallel requests unless explicitly instructed.

This prevents cascading retries that waste tokens and compound the rate limit problem. Once your account matures and limits increase, you can relax the pacing.

Before and After: Real Cost Comparisons

Example 1: Overnight Research Task (6 Hours, 14 Sub-Agents)

A B2B lead research job: find distressed businesses, gather contact info, write personalized outreach.

Setup	What Happens	Cost
All Opus	Every sub-agent runs Opus. 14 agents × 6 hours × full context loading	~$150-200
All Sonnet	Better, but still expensive for research scouts doing simple lookups	~$40-60
Tiered routing	Haiku scouts, Sonnet writer, Ollama file organizer	**$6**

The tiered version used Haiku for the bulk of the work (reading blogs, finding LinkedIn profiles, gathering data). Sonnet stepped in only for crafting the outreach emails. Ollama handled file compilation. 95% of tokens were served from Anthropic's prompt cache, which cut costs further.

Example 2: Daily Personal Assistant

Checking messages, organizing files, answering questions, light scheduling.

Setup	Daily Cost	Monthly Cost
All Opus (default)	$5-10	$150-300
All Sonnet	$2-4	$60-120
Tiered (Haiku primary, Ollama heartbeats)	$0.50-1	$15-30

Example 3: Coding Workflow

Agent assists with code reviews, debugging, feature implementation.

Setup	Daily Cost	Monthly Cost
All Opus	$15-30	$450-900
All Sonnet	$5-10	$150-300
Tiered (Haiku + Sonnet for code, Opus for architecture)	$3-5	$90-150

The Monthly Breakdown

Strategy	Idle Cost	Light Use	Active Use	Heavy Use
Single model (Opus)	$60-150	$150-300	$450-900	$1,500+
Single model (Sonnet)	$30-60	$60-120	$150-300	$500-900
Tiered routing	$0	$15-30	$90-150	$300-500
100% local (Ollama only)	$0	~$2 (electricity)	~$5	~$10

Tiered routing doesn't match the $0/month of running fully local, but you keep the capability of frontier models when you actually need them. For most users, that tradeoff makes sense.

Common Routing Mistakes

Mistake 1: Routing Everything Cheap

Making Ollama or Haiku handle tasks they can't. Your agent tries three times, fails, retries with more context, fails again. You've now spent more tokens on retries than Sonnet would have cost in one pass.

Fix: Be honest about what cheap models can do. Simple reasoning and data extraction, yes. Complex debugging and novel problem-solving, no.

Mistake 2: No Fallback Chain

Running a single primary model with no fallbacks. When Anthropic rate-limits you at 2 AM during an overnight task, the agent stalls until the cooldown expires.

Fix: Always have at least one cross-provider fallback. If your primary is Anthropic, add an OpenAI model as backup.

Mistake 3: Opus for Heartbeats

The default config doesn't separate heartbeat routing. If you set your primary to Opus and forget to set a heartbeat model, you're paying frontier prices 48 times a day for a status check.

Fix: First thing you configure, always: `"heartbeat": { "model": "ollama/llama3.1:8b" }`. See our token optimization guide for the full idle cost breakdown.

Mistake 4: Ignoring Context Bloat

Routing to Haiku saves money per token, but if you're sending 100KB of session history with every request, the savings are smaller than they should be. Context bloat is a multiplier on whatever model you're using.

Fix: Run a "new session" purge before major tasks. Trim context files to under 20KB total.

Getting Started: The 10-Minute Setup

If you're currently running a single model and want to start routing:

Step 1: Install Ollama (2 minutes)

```
curl -fsSL https://ollama.com/install.sh | sh
ollama pull llama3.1:8b
```

Step 2: Add Ollama to your config (2 minutes)

Add the Ollama provider to the `models.providers` section of `~/.openclaw/openclaw.json`.

Step 3: Route heartbeats to Ollama (1 minute)

Add the `heartbeat` block to `agents.defaults`. This alone saves $2-5/day.

Step 4: Switch primary to Haiku (1 minute)

Change `agents.defaults.model.primary` from Sonnet/Opus to Haiku. Add your current model as the first entry in `fallbacks`.

Step 5: Verify (4 minutes)

Watch your agent for a few hours. Check the Anthropic dashboard. Heartbeats should show zero API calls. Most tasks should run on Haiku. Fallbacks should fire only when Haiku gets stuck.

Start here. Add ClawRouter later if you want automatic classification. Add skill-level routing once you know which skills need more intelligence. The two-tier setup (Ollama heartbeats + Haiku primary) captures 80% of the savings with minimal config changes.

Bottom Line

Default OpenClaw wastes 60-80% of your API budget on work that doesn't need frontier intelligence. Heartbeats, file operations, research lookups, data extraction. All of it runs fine on cheaper models or locally.

Three config changes fix this:

1. **Heartbeats → Ollama.** Idle cost drops from $2-5/day to $0.
2. **Primary → Haiku.** 75% of agent work runs at a fraction of Sonnet/Opus pricing.
3. **Fallbacks → Sonnet → Opus.** When Haiku can't handle it, the agent escalates automatically.

The overnight task that cost $150 on Sonnet costs $6 with tiered routing. The daily assistant that drained $10/day drops to under $1. And if you want to skip manual configuration entirely, ClawRouter automates the routing decisions for a 78-99% cost reduction.

Start with the heartbeat fix. It takes one minute and gives you the biggest single savings. Then switch your primary to Haiku. Then add fallbacks. Check your Anthropic dashboard after each change. You'll see the difference immediately.

Related Guides

- OpenClaw Token Optimization — the full 97% cost reduction playbook
- Best Local Models for OpenClaw — which Ollama models handle agent tasks
- Running OpenClaw 100% Local — zero API costs with tradeoffs
- Stop Using Frontier AI for Everything — the general case for tiered routing
- OpenClaw Setup Guide — installation from scratch
- OpenClaw Security Guide — lock down before connecting real accounts
- Local AI Planning Tool — VRAM Calculator

Get notified when we publish new guides.

Subscribe — free, no spam

Source: https://insiderllm.com/guides/openclaw-model-routing/