Module 09

🦞 Advanced Concepts

Cost Optimization, Model Routing, and Power-User Techniques

🧭 Navigation Chart

By the end of this module, you will be able to:

1. **Understand** why 🦞 OpenClaw burns money by default and how to fix it
2. **Implement** the "Brains and Muscles" model for cost optimization
3. **Set up** intelligent model routing (manual, ClawRouter, or OpenRouter)
4. **Enable** prompt caching for up to 90% savings on system prompts
5. **Switch** between AI models for different tasks
6. **Understand** 🐙 multi-agent setups
7. **Use** reverse prompting to let the AI guide your decisions

📜 Ship's Logbook (Part 1)

Term	Definition
Brains and Muscles	Use an expensive model (the "brain") for thinking, and cheaper models (the "muscles") for execution
Model switching	Changing which AI model handles a conversation mid-stream
ClawRouter	🦞 OpenClaw-native routing tool that classifies request complexity and routes to the cheapest capable model
Prompt caching	Provider feature that remembers static parts of your prompt between calls so you pay less

📜 Ship's Logbook (Part 2)

Term	Definition
Context accumulation	Session history growing with every message -- a mature session can reach 200K+ tokens
🐙 Multi-agent	Running more than one 🦞 OpenClaw agent for different life domains
Reverse prompting	Instead of telling the AI what to do, you ask the AI what YOU should do
Open Router	A service providing access to 300+ AI models through a single API
Agentic company	Structuring your AI agents like employees with different roles

The Problem: Why 🦞 OpenClaw Burns Money

Running 🦞 OpenClaw with a single frontier model for all tasks is the **#1 cost mistake**.

Cost Mechanism	What Happens
Context accumulation	Session history grows to 200K+ tokens; every follow-up carries enormous overhead
System prompt re-injection	3,000-14,000 tokens re-sent with every API call
Tool output storage	File listings, browser snapshots bloat context over time
Heartbeat overhead	48 full-context calls/day on Opus = $70-200/month
Cron job overhead	Each trigger creates a fresh conversation with full context

90% of what your agent does is routine work that doesn't need a $5/MTok model.

The Brains and Muscles Model

Split your workload between expensive and cheap models:

The Brain (Opus 4.6)

- Complex decisions and strategy
- Creative writing
- Prompt injection defense
- Orchestrating other models

The Muscles (Cheaper Models)

- Haiku 4.5 -- quick lookups, heartbeats ($1-5/MTok)
- Sonnet 4.5 -- code generation, summarization ($5-15/MTok)
- Codex -- development tasks (varies)

The Opus Orchestra Pattern

A named strategy used by experienced 🦞 OpenClaw operators:

Role	Model	Cost
Orchestration/decisions	Opus 4.6	Premium
Sub-agent execution	Sonnet 4.5	Mid-tier
Heartbeats/routine	Haiku 4.5 or Gemini Flash	Minimal

Real Cost Benchmarks (Community Data)

- **Without optimization:** $100-400/month
- **Well-configured:** $10-50/month
- **Extreme optimization:** <$10/month
- **One user's $800 mistake** -- 8 agents, no limits, pay-per-use

Practical Model Assignments

Task	Recommended Model	Why
Important conversations	Opus 4.6	Best reasoning, most nuanced
Code generation	Sonnet 4.5 or Codex	Optimized for code, cheaper
Heartbeats	Haiku 4.5	Routine checks, very cheap
Web search	Perplexity Pro	Built for search
Document summarization	Sonnet 4.5	Good enough, much cheaper
Quick factual lookups	Haiku 4.5	Fast and cheap

Practical Model Assignments (continued)

Task	Recommended Model	Why
Creative writing	Opus 4.6	Best quality
Security-sensitive tasks	Opus 4.6	Best prompt injection resistance
Budget tasks	Kimi K2.5 (free via NVIDIA)	Comparable to Opus, zero cost
Ultra-cheap tasks	MiniMax M2.5 (~$10-50/mo)	Surprisingly capable; Max plan gives 1,000 prompts per 5 hours
Free tier exploration	Google Cloud ($300 free credits)	Gemini models with sign-up credits (rate limits apply)

Model Switching in the TUI

Switch models during a conversation without starting a new session:

```
/model claude-opus-4-6       # Expensive, best quality
/model claude-sonnet-4-5      # Good balance
/model claude-haiku-4-5       # Fast and cheap
```

When to switch:

- **Opus:** Important decisions, untrusted input, nuanced conversations
- **Sonnet:** Code generation, summarization, drafts
- **Haiku:** Quick facts, formatting, template generation, testing

Intelligent Routing: ClawRouter

ClawRouter is the hottest tool in the 🦞 OpenClaw ecosystem -- 2,400 GitHub stars in its first 11 days.

How it works:

Analyzes each query using a lightweight classifier, then routes to the cheapest capable model.

Tier	Complexity	Routed To	Cost
Simple	Basic lookups, acknowledgments	DeepSeek, Gemini Flash	~$0.27-0.60/MTok
Medium	Moderate tasks, summarization	GPT-4o-mini, Sonnet	~$1-5/MTok
Complex	Analysis, writing, decisions	Claude Sonnet	~$3-15/MTok
Heavy	Multi-step reasoning, agentic tasks	Opus 4.6, Kimi K2.5	~$5-25/MTok

ClawRouter Profiles

Profile	Description
Auto	Balanced quality and cost (recommended starting point)
Eco	Maximum savings, up to 95-100% on simple queries
Premium	Best quality, less aggressive routing
Free	Zero-cost models only (limited capability)

Other routing options:

Approach	Best For
Manual rules in AGENTS.md	Getting started, simple setups
ClawRouter	Most users -- best balance of automation and control
Custom routing 🐠 skill	Power users who want exact control
OpenRouter auto-routing	Users who want zero configuration

Prompt Caching: The Hidden Savings Multiplier

Every API call re-sends your full system prompt (SOUL.md, AGENTS.md, 🪸 MEMORY.md) -- typically **3,000-14,000 tokens**. You pay full price every time.

With caching enabled:

- First call pays full price
- Subsequent calls within the cache window pay **90% less** for cached tokens

Enable it:

```
{
  "cacheRetention": "long",
  "cacheSystemPrompts": true,
  "cacheThresholdTokens": 2048
}
```

- `"long"` uses Anthropic's extended cache (~55 minutes)

The 55-Minute Heartbeat Trick (Revisited)

Combine three optimizations for maximum savings:

1. Set heartbeat interval to **55 minutes** (stays within cache window)
2. Enable **prompt caching**
3. Route heartbeats to **Haiku 4.5**

Without Optimization	With Optimization
Opus for heartbeats	Haiku for heartbeats
Full system prompt every call	Cached system prompt (90% off)
~$100+/month	**~$0.50/month**

99.5% reduction on heartbeat costs.

Prompt caching also works for regular conversations -- 90% off system prompt costs for every

Upgrading Web Search with Perplexity Pro

🦞 OpenClaw's built-in search is functional, but **Perplexity Pro** is significantly better for research.

Setup via Open Router:

1. Create an account at openrouter.ai
2. Get your 🔑 API key
3. Add to 🦞 OpenClaw:

```
openclaw config provider add openrouter --key YOUR_KEY
openclaw config search model perplexity-pro
```

Cost tip:

Save research results as markdown files so you do not pay for the same search twice:

🐙 Agent-to-Agent Communication (sessions_* Tools)

Three built-in tools for agent communication -- no external service required:

Tool	What It Does
`sessions_list`	Discover active sessions and their metadata
`sessions_history`	Fetch transcript logs from another session
`sessions_send`	Message another session; supports reply-back

Example flow: Chief of Staff receives email about crypto → uses `sessions_send` to forward to Crypto Trader → Trader evaluates and sends back assessment → Chief of Staff summarizes for you on Telegram.

Individual agents communicating through `sessions_send` is more reliable and cheaper than one mega-agent doing everything.

🐙 Multi-Agent Management

As you get comfortable, you might want separate agents for different areas of your life.

The Agentic Company Structure:

Real example (7 days, ~$600):

- Sam (Chief of Staff) -- email, calendar, CRM
- Midas (Crypto Trader) -- autonomous portfolio management
- Ritam (Physics Research) -- cross-domain synthesis
- Personal assistant -- daily life management

Individual Agents > Sub-Agents

Community consensus: **multiple individual agents** with separate Telegram bots work better than one agent spinning up sub-agents. Sub-agents lose context and forget. Individual agents maintain their own 🪸 memory.

🚩 Rough Waters: The Coordination Tax

Google DeepMind research: accuracy **saturates or degrades past 4 agents** ("Coordination Tax")

The "17x error trap": more agents = multiplied error rate, not throughput.

The 17 → 4 Consolidation

One user went from 17 agents to 4 core roles:

1. **Architect** (CEO) -- strategy, priorities
2. **Builder** (CTO) -- engineering, quality
3. **Money Maker** -- growth, pricing, channels
4. **Operator** (COO) -- processes, tools, financial ops

Everything else: a **specialist library** (36+ types) spawned on demand by core agents.

Sub-Agent Best Practice

The Freshman Rule & 🐙 Multi-Agent 🪸 Memory

The Freshman Rule

- **One task at a time** -- do not stack requests
- **Ground-up instructions every time** -- assume the agent knows nothing unless it is in files
- **If it is not in files, it does not exist** -- never assume agents remember past sessions

Multi-Agent Memory: 4-Layer Architecture

Layer	What	Example
1: Private	Each agent has its own `MEMORY.md`	Midas knows crypto; Sam knows email
2: Shared dir	A common `_shared/` folder all agents read	User profile, roster, conventions
3: QMD paths	Cross-agent search via shared indexed paths	Any agent searches another's docs
4: Coordinator	Dedicated agent maintaining consistency	Resolves conflicts, propagates updates

The Principle

Reverse Prompting

The normal way (You tell AI what to do):

```
Build me a landing page for my freelance business.
```

The reverse way (AI tells YOU what to do):

```
Based on everything you know about me, my goals,
and my current situation, what should I be working
on right now? What would move the needle the most?
```

Why it is powerful:

- your agent has **perfect 🪸 memory** of every goal and conversation
- your agent has **no ego** -- will tell you uncomfortable truths
- your agent **sees patterns** -- spots contradictions in your behavior

Reverse Prompting: Try These

Strategic:

```
What am I overlooking? Based on my goals and what
I've been working on, what blind spots do you see?
```

Accountability:

```
Review my goals from last month. What did I accomplish?
What did I skip? What pattern do you notice?
```

Self-improvement:

```
Based on our interactions, what are my three biggest
strengths and three biggest weaknesses in how I work?
```

Self-Improvement Workflows

Weekly Agent Tuning (make it a cron job):

Ask your agent:

1. What instructions in your core files feel unclear?
2. What information about me seems outdated?
3. What do I keep asking that should be in your 🪸 memory?
4. What 🐠 skills or automations would make us more efficient?
5. What suggestions do you have for improving our workflow?

Then **actually implement** the suggestions by editing core files.

Context Optimization:

- Use `/compact` to reduce conversation history
- Start new sessions with `/new` for new topics

🚩 Shoals and Sandbars: Chat Quality ≠ Agent Quality

A model that writes beautiful essays may **completely fail** at agentic work. Chat benchmarks do not measure tool-calling reliability, multi-step planning, or error recovery.

Proven for Agentic Work

Model	Strengths
Claude Sonnet 4.5 / Opus 4.6	Reliable tool calls, strong planning, consistent execution
GPT-5.2	Solid multi-step reasoning, dependable function calling
Kimi K2 / K2.5	Comparable to Opus on many tasks, excellent cost-to-quality ratio

Avoid for Autonomous Tasks

Model	Problem
DeepSeek Reasoner	Great at thinking and analysis, but **broken tool calls** -- it reasons beautifully then fails to execute

🚩 Shoals and Sandbars: Common Mistakes

Mistake	Fix
Using Opus for everything	Set up brains and muscles model
Not enabling prompt caching	Enable `cacheSystemPrompts` -- instant 90% savings
Heartbeat interval at 30 min without caching	Set to 55 minutes to stay in cache window
Setting up 5 agents before mastering 1	Master one agent, then expand
Ignoring reverse prompting	Try it once a week
Never doing agent tuning	Schedule weekly tuning (10 min)
Expecting local models to match Opus	Use local for low-stakes tasks, cloud for important work

⚙️ Hands on Deck: Design Your Ideal 🐙 Multi-Agent Setup

Part 1: Map Your Life Domains (10 min)

- List 3-5 areas: personal, professional, business, creative, other

Part 2: Design the Roles (10 min)

For each domain:

- What would the agent focus on?
- What personality should it have?
- What model should it use?

Part 3: Start Simple (5 min)

- Pick the **ONE domain** that would benefit most
- That is your first agent — master it before adding more

💎 Treasure Chest

1. 🦞 **OpenClaw burns money by default** -- five cost mechanisms compound to create huge bills
2. **Brains and Muscles saves serious money** -- expensive model for thinking, cheap for execution
3. **Prompt caching is the biggest hidden savings** -- 90% off system prompt costs
4. **ClawRouter automates routing** -- classifies complexity, routes to cheapest capable model
5. **Model switching is easy** -- `/model [name]` changes mid-conversation
6. **Perplexity Pro is worth the upgrade** -- significantly better research
7. 🐙 **Multi-agent is powerful but complex** -- master one agent first
8. **Reverse prompting is the secret weapon** -- let your agent analyze your situation

🌊 Next Port of Call

Module 10: Security Hardening

Sandboxing with Docker, tool policies, ⛵ gateway hardening, and incident response. Time to lock everything down.