

Claude Model Selection & Task Mapping Guide

A Practical Reference for Cost-Efficient AI Engineering

Original Author: Maribel Hernandez | Reference & Operational Guide

Framework: Capability-to-Cognitive Difficulty

Core Economic Rule: Start with the lowest-cost model that reliably clears the task. Escalate only when reasoning complexity or failure risk demands it. You pay for completed tasks, not raw model power; a 10x cost multiplier on routine code yields 0% quality improvement.

1. Model Tier Overview & Cost Multipliers

Tier	Base Pricing (In / Out per MTok)	Cost for 100k Tokens (50k In / 50k Out)	Relative Cost Multiplier	Best Suited Tasks
HAIKU	\$1 / \$5	\$0.30	1X (BASELINE)	Regex, JSON/YAML formatting, syntax validation, routine transforms
SONNET	\$2 / \$10 (Std: \$3 / \$15)	\$0.60 (Std: \$0.90)	2X TO 3X	High-volume coding, CRUD boilerplate, pipeline automation, sub-agents
OPUS	\$5 / \$25	\$1.50	5X	Multi-file refactoring, deep debugging, architecture design, reviews
FABLE	\$10 / \$50	\$3.00	10X	Persistent autonomous agents, multi-phase migrations, research

2. Task Classification Matrix

Routine Tasks HAIKU Straightforward, predictable, zero advanced reasoning required. - Reformatting 40+ config files (JSON, YAML, CSV) - Generating regex patterns for log parsing and cleanup - Variable renaming and basic string/data transforms - Daily CI/CD static type checks and syntax validation	Repetitive / Mechanical SONNET Patterns repeated across files needing standard coding intelligence. - Generating CRUD API boilerplate for multiple DB tables - Writing unit test suites across similar functions - Updating copyright headers and licenses across codebase - Execution sub-agents in multi-turn workflows
Complex Engineering OPUS Requires cross-file context, deep reasoning, and high reliability. - Redesigning payment reconciliation or core backend services - Debugging race conditions across multi-repo architectures - System architecture design and deep code reviews - Scoping, planning, and critical algorithmic problems	Autonomous Agentic FABLE Long-running sessions needing persistent multi-step coherence. - Multi-phase system-wide platform migrations - Long unattended agentic loops spanning dozens of steps - Autonomous research assistants compiling disparate data - Multi-turn deep edge-case orchestration

3. Smart Routing Framework & Tactical Execution

Check 1: What is the cost of failure?

- **Low / Trivial** (takes seconds to fix, e.g. regex syntax error): Route to **Haiku**.
- **High / Systemic** (risks architecture breakdown or subtle production bugs): Route to **Opus**.

Check 2: How long does the agent run unattended?

- **Single-turn / Fast human check**: Route to **Haiku** or **Sonnet**.
- **Multi-step autonomous loops** (demanding cross-context coherence): Route to **Opus** or **Fable**.

Sub-Agent Optimization

Avoid letting background worker threads inherit expensive frontier rates.

```
export CLAUDE_CODE_SUBAGENT_MODEL=sonnet
```

Automated Split Execution

Plan system architecture with Opus reasoning, implement with Sonnet.

```
claude --model opusplan
```

4. Assessment & Knowledge Verification

Q1: Which task requires escalation to a frontier reasoning tier (Opus/Fable)?

Answer: Redesigning the asynchronous event-driven payment reconciliation system

Routine formatting (YAML), CRUD routes, and header updates are mechanical tasks suited for Haiku or Sonnet. Core architectural redesign requires frontier reasoning.

Q2: Which model tier for daily CI/CD static type checks & syntax validation on 50,000 snippets?

Answer: Claude Haiku

High-volume, deterministic validation tasks should always run on the baseline tier to prevent inflated API spend.

Q3: How to fix excessive billing during multi-repo debugging where parent agent spawns dozens of search sub-agents?

Answer: Export CLAUDE_CODE_SUBAGENT_MODEL=sonnet

Keeps the parent agent on Opus for reasoning while delegating file reads, grep searches, and logs to Sonnet at a fraction of the cost.

Q4: Does claude --model opusplan utilize Opus for planning and automatically switch to Sonnet for execution?

Answer: True

Automates the heavy thinker + fast worker pattern seamlessly.

Q5: Best optimization combination for weekend batch test suite (1,000 internal API endpoints, repeated docs)?

Answer: Route requests through the Message Batches API and enable prompt caching

Batch API delivers 50% cost discount for non-immediate jobs; prompt caching avoids re-billing repeated schemas.

Key Takeaways

- **Evaluate cost per completed task**, not raw model capability.
- **Constrain sub-agents** to Sonnet or Haiku to avoid multiplying expensive frontier token rates.
- **Reserve Opus and Fable** strictly for high-stakes architecture, multi-file planning, and complex debugging.
- **Leverage caching and batch endpoints** for asynchronous and repetitive high-volume workloads.