

CACHE ME IF YOU CAN

CUTTING AI COSTS WITHOUT CUTTING
CORNERS

COLORADO STARTUP WEEK 2025

CHRIS MCDERMUT

THE BILLABLE ~~MINUTES~~ TOKENS PROBLEM

AI CHARGES LIKE A LAWYER – EXCEPT
IT NEVER REMEMBERS YOUR CASE

WHAT'S A TOKEN?

THE CURRENCY OF AI:

- ~4 CHARACTERS = 1 TOKEN
- 1 WORD \approx 1.3 TOKENS
- 1 PAGE OF TEXT \approx 500 TOKENS
- AVERAGE WEBSITE \approx 2,000 TOKENS
- THIS SLIDE \approx 85 TOKENS

GPT-4: \$0.01 PER 1K TOKENS (INPUT) / \$0.03 PER 1K TOKENS (OUTPUT)

THE MATH THAT HURTS

JOB SEEKER (200 JOBS/MONTH):

- 2K INPUT TOKENS * \$0.01 = \$0.02
- 250 OUTPUT TOKENS * \$0.03 = \$0.0075
- PER JOB: ~\$0.03 * 200 = **\$6/MONTH**

REAL ESTATE UNDERWRITER (800 COMPS/MONTH):

- 5 PAGES * 2K TOKENS * \$0.01 = \$0.10 INPUT
- 250 OUTPUT TOKENS * \$0.03 = \$0.0075
- PER COMP: ~\$0.11 * 800 = **\$88/MONTH**

COST CALCULATORS

KNOW BEFORE YOU GO:

- [HTTPS://YOURGPT.AI/TOOLS/OPENAI-AND-OTHER-LLM-API-PRICING-CALCULATOR](https://yourgpt.ai/tools/openai-and-other-llm-api-pricing-calculator)
- OPENAI TOKENIZER → COUNT TOKENS BEFORE SENDING
- TIKTOKEN (PYTHON) → PROGRAMMATIC TOKEN COUNTING

PRO TIP: ALWAYS ESTIMATE COSTS BEFORE DEPLOYING

MONITORING TOOLS

TRACK YOUR ACTUAL SPEND:

- OPENAI USAGE DASHBOARD → DAILY/MONTHLY SPEND
 - HELICONE.AI → ANALYTICS & COST TRACKING
 - LANGSMITH → TOKEN USAGE PER REQUEST
- DIY: LOG {prompt, tokens_used, cost, timestamp}

YOU CAN'T OPTIMIZE WHAT YOU DON'T MEASURE

BEFORE CACHING: THE PAIN

BEFORE CACHING: THE PAIN IN NUMBERS

- > 10 SECOND INFERENCE TIME
 - > \$.01 API CALL

**AFTER CACHING: THE
RESULT**

We're hiring across **four different teams**, each with their own flavor of AI work. You'll work on things like shared libraries, evaluation systems, orchestration patterns, or user-facing features—depending on the team. What they all have in common: you'll ship to production, own meaningful problems, and make an impact across the company.

Due Date

AFTER CACHING: THE RESULT IN NUMBERS

- > MILLISECOND RESPONSE TIME
- > ALL SUBSEQUENT CALLS ARE ESSENTIALLY FREE

THE SOLUTION: THREE-LAYER CACHE

Layer 1: Local Cache (Redis) - \$0.00001/query

↓ (miss)

Layer 2: Vector DB (Pinecone) - \$0.0001/query

↓ (miss)

Layer 3: LLM (GPT-4) - \$0.10/query

MOST QUERIES NEVER REACH LAYER 3

STRATEGY 1: EXACT MATCH CACHE

HOW IT WORKS:

- USER ANALYZES SAME DOCUMENT + PROMPT
 - CACHE KEY: doc_id + prompt
- 'JOBLISTING123 + SUMMARIZE' → CACHED RESULT
 - HIT = RETURN INSTANTLY (<1MS)

BEST FOR: REPEATED DOCUMENT ANALYSIS, SAME REPORTS

STRATEGY 2: PRE-HYDRATION

PROACTIVE CACHING (USING CHEAPER MODELS):

1. ANALYZE YOUR LOGS → FIND TOP QUERIES
2. GENERATE RESPONSES DURING OFF-PEAK HOURS
3. USE CHEAPER MODELS (GPT-3.5) FOR BATCH PROCESSING
4. LOAD CACHE BEFORE USERS WAKE UP
5. FIRST-TIME USERS GET INSTANT RESPONSES

BEST FOR: PREDICTABLE QUESTIONS, ONBOARDING, DEMOS

STRATEGY 3: VECTOR SIMILARITY CACHE

HOW IT WORKS:

- USER ASKS: 'HOW DO I CHANGE MY PASSWORD?'
- CONVERT TO VECTOR EMBEDDING (~\$0.0001)
- SEARCH FOR SIMILAR VECTORS (>85% MATCH)
- 'PASSWORD RESET' = 94% SIMILAR → RETURN CACHED

BEST FOR: VARIATIONS OF SAME INTENT

REAL CODE (SIMPLIFIED)

```
class SmartCache:
    def get_response(self, query):
        # Try Redis first
        if cached := self.redis.get(query):
            return cached

        # Try vector similarity
        embedding = get_embedding(query)
        if similar := self.vector_db.query(embedding, threshold=0.85):
            return similar.response

        # Last resort: expensive LLM
        return call_llm(query)
```

THE PROJECTED RESULTS

BEFORE: \$1,650/MONTH. 3-5 SECOND RESPONSES

AFTER: \$165/MONTH. <100MS RESPONSES

90% COST REDUCTION

50X FASTER

COMMON PITFALLS

1. MEASURE BEFORE OPTIMIZING

2. CHOOSE THE RIGHT TTL

3. START WITH 0.8 SIMILARITY THRESHOLD

4. VERSION YOUR CACHE ON MODELS/PROMPTS/CONTEXT
UPDATES

QUICK WINS YOU CAN DO TODAY

1. LOG YOUR LLM CALLS (30 MINS)
2. FIND YOUR TOP 10 QUERIES (1 HOUR)
3. CACHE THOSE 10 (2 HOURS)
4. SAVE 30% BY TOMORROW

LET'S CONNECT

I'M BUILDING THIS NOW. WANT TO COMPARE NOTES?

EMAIL: CHRISMCDERMUT@GMAIL.COM

PROJECTS: [CAREEROPS.COM](https://careerops.com) + [USEINFLOW.IO](https://useinflow.io)

GITHUB: [CHRISMCDERMUT](https://github.com/CHRISMCDERMUT)

WEBSITE: CHRISMCDERMUT.COM

LINKEDIN: [/IN/CHRISMCDERMUT](https://www.linkedin.com/in/CHRISMCDERMUT)

Q&A TIME

RESOURCES

HENRY SHI'S AI CRASH COURSE [HTTPS://GITHUB.COM/
HENRYTHE9TH/AI-CRASH-COURSE](https://github.com/HenryThe9th/AI-Crash-Course)

BOOK: ALEX XU'S SYSTEM DESIGN INTERVIEW I + II

LEARNING PLATFORM: [PLURALSIGHT.COM](https://www.pluralsight.com) + [TRYSEXPONENT.COM](https://trysexponent.com)

CERTS: AWS DEVELOPER AWS SOLUTIONS ARCHITECT

**REMEMBER
EVERY CACHED QUERY IS
RUNWAY EXTENDED**

THANK YOU!