4× Claude Code Max hala yetmiyor — gerçekten işe yarayan şeyler bunlar

Caveman, context compression ve feed'inin sürekli önüne attığı 'token tasarrufu' plugin'larının bir power user değerlendirmesi. Çoğu faturanın en ucuz dilimini optimize ediyor.

Yiğit Konur · 21 Nis 2026 · 9 dk okuma

▶ Ana fikirler

dört adet 20× Claude Code Max aboneliği paralel çalıştırıyorum. default kotanın yaklaşık 80 katı. yine de duvara çarpıyorum.

egzotik bir şey yaptığım için değil. ciddi production hacminde Claude Code, üzerine para atabildiğinden daha hızlı token yakıyor. abonelik sınırı aslında bir "sınır" değil — iyi orchestrate edilmiş bir agent'ın LLM'e context yükleme hızı. belirli bir iş yükü eşiğinin ötesinde, her şeyi uçtan uca yapan tek bir Claude Code instance bottleneck haline geliyor.

o yüzden gossip'i yazdım — **Claude planlar**, **Codex çalıştırır**, ikisi yapılandırılmış bir channel üzerinden birbirleriyle konuşur. Claude mimar, Codex işçi. tek bir provider'da dikey scale değil, provider'lar arasında yatay scale.

ama oraya gelmeden önce çoğu kişi token tavanına çarpar ve Google'a koşar. **caveman** bulur. **context compression plugin'ları** bulur. %75–95 tasarruf vaat eden yirmi tane Medium yazısı bulur. ve hepsini yükler.

bu yazının dürüstçe cevaplamaya çalıştığı soru şu:

İçeriğe geç

> *bu "token tasarrufu" şeylerinin hangisi gerçek, hangisi tuzak — ve Claude Code*

> *workflow'unu gerçekten production yüküne nasıl dayanıklı hale getiriyorsun?*

bir gün harcadım: her önemli Reddit thread'ini, bulabildiğim her benchmark'ı, gerçek caveman kaynak kodunu ve Anthropic dokümanlarını okudum. işte durum bu.

caveman gerçekte ne

caveman, Julius Brussee'nin yazdığı bir Claude Code skill'i. 14k yıldız. söz verdiği şey: Claude'un output'undaki "Certainly! I'd be happy to help..." gibi dolgu cümlelerini kazımak. kodu bırak, girişi at.

teknik olarak şunu yapıyor: session başlangıcında bir sistem kuralı inject ediyor ve Claude'a sıkıştırılmış, fragment ağırlıklı yazmasını söylüyor. artikel yok, çekince yok, nezaket lafları yok. üç yoğunluk seviyesi (`lite`, `full`, `ultra`). code block'lar, dosya yolları, commit mesajları, tool call'lar — hepsi dokunulmaz. güvenlik uyarıları ve belirsizliğin tehlikeli olduğu her şey için otomatik kapanıyor.

kurulum tek satır:

```
npx skills add JuliusBrussee/caveman
```

README ~%75 output-token azalması iddia ediyor. bu sayı işin ilginçleştiği yer.

README'nin göstermediği matematik

viral thread'lerde kimsenin söylemediği şey şu: **caveman sadece output token'larına dokunuyor**. gerçek bir Claude Code session'ında output faturanın küçük dilimi.

bulduğum en temiz analiz Mejba'nın — gerçek bir session'ı gerçekten ölçmüş. kabaca:

bu		session başına token	caveman ne yapıyor
inp	ol def'ler	~15.000	hiçbir şey
input: conversation history (her turda yeniden okunan)		~35.000	hiçbir şey

İçeriğe geç

bucket	session başına token	caveman ne yapıyor
input: Claude'un context'e çektiği dosyalar	~25.000	hiçbir şey
output: Claude'un prose + tool call'ları	~25.000	prose kısmının ~%75'ini kesiyor
toplam	**~100.000**	**~4.500 token tasarruf**

bu **gerçek faturada ~%4,5 azalma** demek, %75 değil. ağır API kullanımındaysan belki aylık 15–20 dolar. güzel. devrim değil.

caveman'ın yazarı, kendine kredi verelim, Hacker News'te bunu kabul etti: %75 rakamı ön testlerden geliyordu, titiz bir benchmark'tan değil; skill zaten gizli reasoning token'larını azaltmak için tasarlanmamıştı.

geri kalan %95'in nereye gittiği:

```
┌──────────────────────────────────────────────┐
│  CLAUDE CODE TOKEN BURN — REAL DISTRIBUTION  │
├──────────────────────────────────────────────┤
│  repo exploration / file scanning      ~35%  │  ← biggest sink
│  conversation history re-reads         ~25%  │  ← compounds every turn
│  MCPs + skills loaded into context     ~15%  │  ← quietly brutal
│  extended thinking / reasoning         ~15%  │  ← the real expense
│  output prose (caveman hits this)      ~10%  │  ← the small slice
└──────────────────────────────────────────────┘
```

Reddit aslında ne düşünüyor

içeri girerken tipik bir cargo-cult tapınması bekliyordum. bulduğum şey yüzey hype yazılarının altında oldukça ayakları yere basan bir topluluktu.

r/ClaudeCode: *"does caveman plugin really help with context usage?"*

küçük thread, 14 yorum, ama sinyal sıkı. **u/ConnectTransition660**'ın en üstteki cevabı ge İçeriğe geç ı aktarıyordu — pratikte yaklaşık %30 tasarruf, %75 değil. README'nin ha

en çok upvote alan eleştirel yorum, **u/Kaskote**'dan:

> *cool idea, but this optimizes the cheapest part of the bill.*

o tek satır analizin tamamı. output token'lar ucuz kısım. input context — repo'lar, history, tool schema'ları — paranın gerçekte gittiği yer; caveman bunların hiçbirine dokunmuyor.

u/Revolutionary-Tough7 diğer kilit içgörüyü bıraktı:

> *it's not the prompts that cost the money. it's the thinking.*

abonelik planlarında 5 saatlik pencerede ~19 milyon token alıyorsun. output prose'dan birkaç bin token kazanmak bu ibreyi oynamıyor. ihtiyaç duymadığında extended thinking'i kapatmak oynuyor.

r/ClaudeAI: *"taught Claude to talk like a caveman to use 75% less tokens"*

caveman'ı haritaya koyan viral post bu. 12,6k upvote, 581 yorum. en üstteki yorum projenin tamamını anlatan şakayla **u/fidju**'dan:

> *why waste time say lot word when few word do trick?*

sadece bu 12,4k upvote aldı. ama meme katmanının altında ciddi eleştiriler de karşılık buldu. yüksek puanlı ciddi cevaplardan biri:

> *forcing Claude to talk like a caveman might actually make it dumber.*

argüman şu: modeli "daha az zeki bir persona"ya zorlarken prose'la birlikte reasoning kalitesini de düşürebilirsin. kulağa mantıklı geliyor. doğru mu? gerçek benchmark'lara dayanan kısa cevap: **hayır**. Mejba'nın yan yana testleri caveman moduyla first-attempt başarı oranlarının hafifçe *arttığını* gösterdi (64% → 71%); Mart 2026 tarihli bir arXiv ma[İçeriğe geç]a yanıtları zorlamanın büyük modellerde bazı benchmark'larda doğruluğu 26 yüzde puanına kadar artırabildiğini buldu. mantığa aykırı ama gerçek bir etki var — verbose default'lar fluff-as-reasoning'i teşvik ediyor gibi.

r/ClaudeCode: *“I saved $60 by building this tool to reduce Claude Code token usage”*

konuşmanın daha olgunlaştığı yer burası. tool, Claude’un her task’ta repo’nu yeniden keşfetmesini engelleyen bir pre-indexing katmanı. yazarın benchmark’ı **%54 daha az token** gösterdi; yorumlar da büyük ölçüde anlaştı: asıl israf prose değil, repo exploration.

bu thread ve Kilo Code tartışmasında tekrarlayan bir yorum kalıbı: **CLI output gizli katil**. test runner’lar, derleyiciler, linter’lar, dev server’lar — hepsi verbose output fışkırtıyor ve bu çıktı LLM’e olduğu gibi besleniyor. bir thread model’e ulaşmadan önce CLI noise’u filtreleyerek iki haftada 10 milyon token tasarruf edildiğini aktardı. dar ama yaygın bir workflow için ~%89 tasarruf bu.

r/ClaudeCode: *“don’t use Claude Code’s default system prompt”*

farklı bir açı: plugin ekosistemini tamamen geç, system prompt’u `--system-prompt` ile override et ve kendi kurallarını 500 token’ın altında tut. consensus şu: **CLAUDE.md zaten çoğu workflow için önemli olan şeylerin %90’ını yapıyor**, default system prompt herkese hizmet etmeye çalıştığı için şişirilmiş durumda.

u/AgreeableFall5530 — kurulum pitchi ile dürüst matematiği birleştiren bir yorum:

> *75% is not realistic for normal English in my experience.*

devamında önerdiği şeyler (kısa CLAUDE.md, MCPs’i söküp CLI flow’larıyla değiştirmek, büyük log yapıştırmaktan kaçınmak, hook tabanlı PDF-to-markdown dönüşümü) caveman pitchinden daha fazla upvote aldı. topluluk, dikkatli okuyunca, viral içeriğin bir adım önünde zaten.

token tasarrufu sağlayan şeylerin gerçek hiyerarşisi

Re İçeriğe geç ı ve benchmark’lar bir konuda hemfikirse bu şu: **caveman iyi ama lis d’lerin ve ölçümlerin gerçekten desteklediği şeylere dayanan etki-emek sıralaması:

sıra	müdahale	efor	gerçekçi tasarruf	notlar
1	rutin task'larda extended thinking'i kapat	1 dk	%10–20	Reddit'in en hafife alınan kolu. mimari işler için tekrar aç.
2	kullanılmayan MCPs + skill'leri denetle ve sil	30 dk	%15–25	bazı kişilerin 160+ kayıtlı skill'i var. her biri her çağrıya vergi kesiyor.
3	repo'nu önceden index'le (ai-codex, GrapeRoot, ContextKing, Serena)	15 dk	exploration ağırlıklı işlerde %30–50	Claude'un her task'ta codebase'ini grep'lemesini durduruyor.
4	CLI output'unu context'e girmeden önce filtrele (RTK, Headroom)	20 dk	test/build döngüleri için %89'a kadar	`npm test` döngüsü çalıştıran herkes için gizli katil.
5	ilgisiz task'lar için yeni session başlat	0 dk	%10–15	tek satırlık bir düzeltme için devasa bir session'a chat-continue yapma.
6	basit task'lar için `/model haiku` kullan, gerektiğinde opus	0 dk	maliyette %20–40 (token sayısında değil)	routing compression'dan ucuz.
7	caveman plugin'ı	5 dk	toplamda ~%4–5	komik, zararsız, marjinal. kur, geç.
8	kısa CLAUDE.md ile özlü direktifler	5 dk	%5–10	"be concise. no filler. conclusions first." — caveman'ın %80'ini bedavaya yapıyor.
9	custom `--system-prompt` override	20 dk	değişken, çoğunlukla davranışsal	token'dan çok kaliteyle ilgili.

caveman tuzak değil. çalışıyor, bedava, 5 dakikalık kurulum. sadece production hacminde duvara çarparsan seni kurtaracak şey bu değil. yukarıdaki liste gerçekten fark yaratacak şeylerin kabaca sırasında.

bunların hiçbiri yetmediğinde

gerçek hacimde Claude Code çalıştıran herkes için rahatsız edici gerçek şu: **token optimizasyon plugin'ları ciddi bir workflow'un yüküne kıyasla yuvarlama hatası.**

paralel birden fazla coding agent çalıştırıyorsan, her gün feature gönderiyorsan, aynı pipeline'da araştırma + refactor + review yapıyorsan — tek abonelik yetmeyecek; var olan her caveman tarzı plugin'ı üst üste yığmak bunu değiştirmiyor. optimizasyonlarla belki %30–40 daha fazla nefes alanı açabilirsin. prose üzerinde zekice olmakla throughput'u 10 katına çıkaramazsın.

o ö İçeriğe geç kten işe yarayan şey **yatay scaling**:

1. ayrı workload'larda paralel çalışan birden fazla abonelik. orchestrate etmesi can sıkıcı ama gerçek.
2. planlama ve çalıştırmayı farklı model/provider'lara böl. planlama ucuz, çalıştırma pahalı. pahalı modelin daha az düşünmesine izin ver.
3. gürültülü işi daha ucuz agent'lara devret. test output'unu ana agent'ın context'ine ulaşmadan önce Haiku seviyesi bir modele özetle. Claude denetlerken Codex kaba düzenleme yapsın. her modelin iyi olduğu şeyi yap.
4. agresif cache'le ve cache'i bozan hamlelerden kaçın. konuşmanın ortasında model değiştirmek, thinking ayarlarını açıp kapatmak, tool listelerini yeniden sıralamak — bunların hepsi prompt caching'i geçersiz kılıp tüm session history'nin maliyetini yeniden üstüne yükleyebilir.

gossip'i yazmamın sebebi temelde bu — Claude planlar, Codex çalıştırır, yapılandırılmış bir channel üzerinden iletişim kuruyorlar. caveman kötü olduğu için değil. caveman'ın çözdüğü problem bu tür bir iş yükü için yanlış irtifada olduğu için.

tek ekranda aksiyon planı

her şeyi hızlıca gezdiysen, sırayla yapılacaklar:

İçeriğe geç

```
WEEK 1 — free wins, zero risk
├─ [ ] turn off extended thinking by default (huge, underrated)
├─ [ ] run /doctor, audit installed skills and MCPs, remove anything unused
├─ [ ] add 4 lines to CLAUDE.md: "be concise. no filler. no hedging.
│      conclusions first. skip pleasantries."
├─ [ ] start a new session for any task that isn't a direct continuation
└─ [ ] stop changing models mid-conversation (cache-busts everything)

WEEK 2 — light tooling
├─ [ ] install a repo pre-indexer (ai-codex, Serena, ContextKing)
├─ [ ] if you run tests/builds in loops, add a CLI output filter
├─ [ ] install caveman if you want the joke — it does help a little
└─ [ ] measure with /usage before and after every change

WEEK 3 — structural
├─ [ ] if still hitting limits, look at horizontal scaling —
│      multiple subs, multi-provider orchestration
├─ [ ] split planning vs execution across models
├─ [ ] consider moving the noisy stuff off-agent entirely
└─ [ ] only now is caveman's 4-5% actually worth optimizing for
```

tldr

caveman akıllıca bir skill. eğlenceli. söylediği gibi çalışıyor — *hedeflediği spesifik şeyde*. sorun şu: hedeflediği şey faturanın en ucuz dilimi, viral içerik ise başka şeyi ima etti.

Reddit topluluğu, meme yorumlarının ötesini okuyunca, bunu zaten biliyor. ciddi yorumlar hep aynı birkaç gerçek kola işaret ediyor: ihtiyaç duymadığında extended thinking'i kapat, hiç kullanmadığın 100 skill'i yüklemeyi bırak, repo'nu önceden index'le, CLI noise'u filtrele, yeni session başlat. bunlar sana %4 değil %50+ nefes alanı kazandıran şeyler.

ve tüm bunları bir arada yapsan bile yetmeyecek bir ölçekteysen — welcome to the club. plugin'lerinle çıkış bulamazsın. mimarınla çıkış bulursun. birden fazla hesap, bir İçeriğe geç del, akıllı orchestration. gerçek nefes alanı orada.

caveman'ı kur. biraz gül. sonra gerçek işe dön.

Kaynaklar

1. caveman by Julius Brussee
2. Mejba's caveman teardown — real token instrumentation
3. r/ClaudeCode: Does caveman plugin really help with context usage?
4. r/ClaudeAI: Taught Claude to talk like a caveman to use 75% less tokens
5. r/ClaudeCode: I saved $60 by building this tool to reduce Claude Code token usage
6. r/ClaudeCode: Don't use Claude Code's Default System Prompt
7. arXiv: Brevity constraints and reasoning accuracy in large models (March 2026)
8. gossip — Claude plans, Codex executes

Sorular ve yanıtlar

▶ Yani caveman bir tuzak mı?

▶ En büyük kazancı sağlayan tek değişiklik hangisi?

▶ Claude'u kısa konuşmaya zorlamak onu daha mı aptallaştırıyor?

▶ Plugin stack ne zaman işe yaramaz?

✳

Bir sonraki merak yolculuğunda görüşürüz.

Diğer notlar →