Token Saver: Cost Savings for Pay-Per-Token Buyers, Usage Time for Subscription Buyers

Internal study — the 80% cost headline for pay-per-token buyers and the measured 1.8x token ratio turned into usage-time for subscription buyers, with the fee and allowance terms.

Generated: 2026-10-03 18:35 UTC

Token Saver: Cost Savings for Pay-Per-Token Buyers, Usage Time for Subscription Buyers

Internal study — October 3, 2026

Version: 1.0
Last updated: 2026-10-03

Abstract

Token Saver is bought two different ways, and one number cannot honestly describe both. A pay-per-token buyer sees a dollar bill, so the relevant result is a **cost saving**. A subscription buyer pays a fixed price for an allowance and never sees a per-token bill, so the relevant result is a **usage-time increase**: how much longer the same plan lasts. This paper states each result in the unit that fits its buyer. For pay-per-token buyers, one real production account paid 80% less over 30 days than the same work priced entirely on a frontier flagship. For subscription buyers, our planning-time token ratio of about 1.8x stretches a 5-hour window to roughly 9 hours of equivalent work, on hosts that meter tokens. Both figures are reported, not projected, and both carry the limits listed below. Derivations that already exist in the Token Saver cost whitepaper are cross-referenced here rather than repeated.

1. Cost savings (pay-per-token buyers)

1.1 The headline: 80%

The figure we publish is **80%**. It was measured on one real production account over 30 days: $2,323 when priced solely on a frontier flagship, against $463 right-sized. It is reported, not projected, and it is a deliberate underpromise (`tokensaver-site/lib/site-stats.ts` , `pricedBothWays`). It describes one account. It is not a guarantee for any other account.

1.2 Other figures, and exactly what each one is

- **93.86%, in our own studies, on one case.** This is a whole-platform measurement over 18,870 runs against a frontier-market-average baseline (`tokensaver-site/lib/proof-stats.ts`). It is a result from our own studies on that one case. It is never a promise and is not the headline.
- **56.7%.** The combined cross-platform benchmark: our four-complexity benchmark, re-run end to end inside Claude and then inside ChatGPT. Always quote it with that provenance.
- **83%.** The four-complexity benchmark cut cost 83% at held quality on our own platform, every test case cheaper. It is a benchmark result on our own platform, not a platform-wide savings claim.

1.3 What the buyer actually keeps

Token Saver charges a fee of **20% of dollar savings** (`TOKEN_SAVER_FEE_FRACTION` , `backend/billing.py:136-180`). A fee-free allowance of **500K saved tokens per UTC day** applies to anonymous, signed-in free, and non-paying-organization principals. The allowance is **0** for any principal on an active or trialing paid subscription, user or organization, because the paid plan already carries its own credits (`_allowance_for` in `backend/services/token_saver.py:7164-7188` ; `resolve_paid_program_enrollment` , same file, line 6668). In practice:

- Free users keep 100% of their savings up to the daily cap.
- Paid users keep 80% of every saved dollar.
- The fee is identical across Starter, Standard, Pro, and Organization plans by design, so Token Saver savings do not vary with the plan tier.

The 20% fee behaviour passed its tests on 2026-10-03 (257 tests).

1.4 Illustrative monthly scenarios

This table is a scenario, not a measurement. It assumes $0.1162 of gross savings per 1K saved tokens, derived from two live planner quotes on 2026-10-03, over 30 days.

Usage	Plans/day	Saved tokens/plan	Gross savings	Fee (20%)	Net, paid plan	Net, free (within cap)
Light	5	136	$2.37	$0.47	$1.90	$2.37
Typical	25	163	$14.21	$2.84	$11.36	$14.21
Heavy	100	190	$66.23	$13.25	$52.99	$66.23

Real savings depend on the host, the task mix, and which model tier the work would otherwise have run on.

2. Usage-time increase (subscription buyers)

2.1 Why the dollar figure does not transfer

A dollar ratio prices the same tokens two ways. Against a fixed-price allowance it means nothing. The unit that transfers is the **token ratio**: allowances are metered in tokens or a token-weighted unit, so fewer tokens for the same finished work means less allowance consumed. The derivation, and the caveats attached to it, are in the "Quota runway" section of the cost whitepaper. This paper reuses that measurement and adds the time arithmetic.

2.2 The measurement

Across 508 of Elis's own production plans, the same finished work took 1,361,307 baseline tokens against 751,419 optimised: **44.8% fewer tokens, 1.81x the output per allowance**, published as "about 1.8x". The distribution is p10 1.50x, p50 1.73x, p90 2.53x (`backend/agents/seeds/seed_pitch_copilot.py:405-419`; the same figure appears in the cost whitepaper).

2.3 Time-extension arithmetic (new in this paper)

On a host whose plan limit is metered in tokens or a token-weighted unit over a rolling window, a multiplier *m* stretches a window of *W* hours of work to *m* x *W*.

Point in distribution	Multiplier	A 5-hour window holds
p10	1.50x	7.5 hours of equivalent work
p50	1.73x	8.7 hours
Mean	1.81x	9.1 hours
p90	2.53x	12.6 hours

Weekly cap. Take a developer whose weekly cap runs out on Thursday (day 4 of 7). At p10 the cap lasts to about day 6 (Saturday). At p50 it lasts to about day 6.9, just short of the full week. At the mean and at p90 the week does not run out at all (4 x 1.81 = 7.24, which exceeds 7). The line already published, "reaches Friday evening", sits below p10 and is a deliberate underpromise.

Team framing. At the mean multiplier, 100 seats produce the output of roughly 180.

2.4 What does not change

Wall-clock time per task is unchanged. Per-minute rate limits are unchanged. The host's window lengths are unchanged. This is more work per window, not faster work.

2.5 Which hosts this applies to

As of October 2026, and drawn from third-party summaries (readers should verify against each vendor's official documentation):

- **Claude Pro/Max and ChatGPT/Codex** meter a 5-hour rolling window plus weekly caps in a token-weighted unit. The token ratio applies.
- **GitHub Copilot** (premium requests per month) and **Cursor** (credits) meter requests. Token savings do not extend those allowances. Only routing a request to a lower-multiplier model does, and this paper does not price those multipliers.

3. Methodology

Cost figures in Section 1 come from production accounting priced two ways over the same work (the 80% account) and from the whole-platform and benchmark runs named above. Subscription figures in Section 2 come from the planning-time `estimated_baseline_tokens` and `estimated_optimized_tokens` recorded on every production plan. The time arithmetic is multiplication, with no further modelling. The scenario table in 1.4 is arithmetic on stated inputs.

4. Limitations

- **Small, internal sample.** The 508 plans come from 3 internal users on short tasks.
- **Planning-time estimator.** The token ratio is Elis's own baseline-versus-optimised estimate at planning time. It is not usage reported back by a host.
- **Not broken out per host.** The ratio is aggregated across hosts and tasks. Direction differs by host: the ChatGPT benchmark increased visible tokens 40.7% while cutting cost 36.3%, whereas the Claude benchmark cut tokens 28.9%. On a host whose decomposition inflates tokens, an allowance can go further on cost and less far on quota in the same session.
- **Weighted-quota upside and cache reads are deliberately not priced.** No provider publishes its quota weighting, and how a meter treats a cached read is undocumented. This paper publishes no cost-weighted multiplier.
- **One account behind 80%.** The headline is one account over 30 days.
- **Scenario inputs are two quotes.** The 1.4 table rests on two live planner quotes from a single day.
- **Next step.** A pre-registered matched-week field study, the same user routed through Elis for one week and not for another under an identical allowance, is the registered way to turn the planning-time estimate into a measured runway.

5. What this paper does and does not claim

It claims: one production account paid 80% less over 30 days; a published token ratio of about 1.8x over 508 internal plans; and arithmetic showing what that ratio means for a 5-hour window and a weekly cap on token-metered hosts.

It does not claim: that 80%, 93.86%, 56.7%, or 83% will repeat for any other account; that usage time extends on request-metered hosts; that tasks finish faster; that the 1.8x figure is a measured floor or is host-independent; or any cost-weighted quota multiplier.

6. Sources

- `tokensaver-site/lib/site-stats.ts` (`pricedBothWays` : $2,323 versus $463, 30 days)

- `tokensaver-site/lib/proof-stats.ts` (18,870 runs; own-studies figure)
- `backend/billing.py:136-180` (`TOKEN_SAVER_FEE_FRACTION`)
- `backend/services/token_saver.py:7164-7188` (`_allowance_for`) and `:6668` (`resolve_paid_program_enrollment`)
- `backend/agents/seeds/seed_pitch_copilot.py:405-419` (508-plan token ratio)
- `frontend/public/docs/whitepapers/elis-token-saver-cost-whitepaper.md` ("Quota runway" section)
- `frontend/public/docs/whitepapers/co2-savings-right-sizing.md` (format and right-sizing background)
- Public site: https://elistokensaver.com

© 2026 Elis AI. All rights reserved. tryelisai.com