# Nanoblog Index: Budgeted Parallel Lookup and Tweet-Count Ranking of Verified Humans on X

Shyamal Chandra
Nanoblog Index
Private ranking worker and public observatory

***Abstract*—X publishes no global catalog of accounts and no lifetime ranking by post volume. Nanoblog Index builds two public boards—celebrity and civilian—each capped at $100{,}000$ seats, and ranks verified humans by raw lifetime `tweet_count` only. The private TypeScript worker looks up seed handles through the official X API in batches of $100$, then writes a JSON snapshot the public Next.js site can render without holding a bearer token. This paper specifies the frozen grade covenant, the human filter, the seed-overlay snapshot, and a pre-calculated request budget that fans remaining lookups across at most eight parallel workers after a single probe call. We report local runtime, test-suite cost, and the request arithmetic for the current $2{,}533$-handle seed. The design treats HTTP 402 (credits depleted) as a hard stop, not as evidence that zero accounts exist.**

***Index Terms*—social ranking, X API, tweet count, budgeted parallelism, verified humans, software architecture**

## I. Introduction

Public attention metrics on X (formerly Twitter) are usually engagement graphs: likes, reposts, impressions, or follower counts. Those signals are easy to game, often windowed rather than lifetime, and in the current app-only API they are not even returned as lifetime totals for every account. What the official user object *does* return is `public_metrics.tweet_count`: a raw lifetime post total [1].

Nanoblog Index treats that integer as the only ranking key. Two lists are published side by side. A celebrity seat is $1{,}000{,}000$ or more followers, or a handle on a locked curated set. Everyone else who passes the verified-human filter sits on the civilian list. Each list has $100{,}000$ seats. Machines, business profiles, and API-generated accounts are excluded [2], [3].

The product is split on purpose [4]. A private worker owns the X SDK and the bearer token. It writes `data/leaderboard.json` and SVG figures. A public Next.js 15 site [5] reads the seed file plus that snapshot and never imports `@xdevplatform/xdk` [6].

X still does not offer "top accounts worldwide." User search needs a user OAuth token [7], [8]. Recent post search needs a positive keyword and spends the same credit pool. A full walk of the handle language $[A-Za-z0-9\_]^{1..15}$ has size $37^{15}$ [9]. The operational path is therefore a curated seed, optional bounded discovery, and a request budget computed before the second network call.

This paper is the industry specification for that path. Section II states the API and ranking background. Section III gives the ranking, filter, snapshot, and budgeted worker algorithms. Section IV states complexity and wall-clock bounds. Section V reports local measurements. Section VI lists work that is still out of reach. Section VII concludes.

## II. Background

### A. Why tweet count, not graph rank

Classic web ranking uses hyperlink graphs. PageRank iterates a random walk on the citation graph [10]. HITS separates hubs from authorities [11]. Those methods need a crawlable edge set. X does not give this worker a global follow graph, and lifetime likes, reposts, replies, and impressions are not on the user lookup object. Using them would require inventing numbers. The index therefore sorts by $t(u) = \textit{tweet\_count}(u)$ and breaks ties by username [12].

### B. Bots and non-human accounts

Automated accounts have been a first-class research object for more than a decade [2], [3], [13]. Nanoblog Index does not claim a trained classifier. It applies a conservative, inspectable filter: the account must be `verified === true`; `verified_type` must not be `business` or `none`; the handle must not look like a bot or API account; the biography must not advertise automation or LLM generation. False negatives (quiet humans) stay on the board as Spark. False positives (clever bots) are a known residual risk.

### C. API topology and credits

App-only bearer tokens can call `GET /2/users/by` with up to $100$ usernames [1], [6]. `GET /2/users/search` requires a user OAuth token [7]. `GET /2/tweets/search/recent` accepts a bearer token but rejects standalone operator-only queries and burns credits. When the project is out of credits, both lookup and recent search return HTTP 402 [14]. The worker maps 402, 429, 401/403, and 400 to typed stop reasons and retries only transient connect failures (three attempts, $800\,\mathrm{ms}$ base delay) against `api.x.com` then `api.twitter.com` [15].

### D. Product split

The public site must not depend on the X SDK. Seed handles live in `data/seed-accounts.txt` under `# --- Celebrity` and `# --- Civilian`. Live metrics overlay matching rows. Placeholder rows keep `tweet_count`=0 so a credit outage cannot empty the observatory. Fixture handles `star_*` and `voice_*` never appear on the public boards.

TABLE I
FROZEN GRADE FLOORS (COVENANT V1)

| Roman | Grade | Floor $t(u)$ |
|---|---|---|
| I | Mythic | 1,000,000 |
| II | Titan | 500,000 |
| III | Oracle | 250,000 |
| IV | Sage | 100,000 |
| V | Scribe | 50,000 |
| VI | Voice | 20,000 |
| VII | Signal | 5,000 |
| VIII | Ember | 1,000 |
| IX | Spark | 0 |

**Input:** users $U$, seat cap $k$
**Output:** rows with $\mathrm{rank} \in 1..|R|$
$R \leftarrow \{\, (u,\, t(u)) \mid u \in U \,\}$
sort $R$ by $t$ descending, then username ascending
**return** first $\min(k, |R|)$ rows numbered from 1

**Algorithm 1:** RankByTweetCount

## III. ALGORITHM

### A. Frozen covenant v1

Grade floors are locked and never retuned to flatter a reshuffle.

List classification is

$$\mathrm{list}(u) = \begin{cases} \text{celebrity} & f(u) \geq 10^6 \text{ or } u \in C, \\ \text{civilian} & \text{otherwise,} \end{cases}$$

where $f(u)$ is followers and $C$ is the curated handle set.

The visual tween is not an eligibility cut. Let $F = 1{,}000$ and $T = \max_u t(u)$ on that list ($T = 0$ if the list is empty).

$$\mathrm{tween}(t, T) = \begin{cases} 0 & T \leq 0 \text{ or } t \leq F \text{ or } T \leq F, \\ 1 & t \geq T, \\ (t - F)/(T - F) & \text{otherwise.} \end{cases}$$

A seed board with every $t(u) = 0$ still seats every handle. The number-one row is always tween 1 once a positive maximum exists.

### B. Rank

Each list is ranked independently, then a global rank is assigned across the concatenation. Engagement columns stay 0 with `engagementSource=unavailable`. `humanRawTotal` equals tweet count in that state.

### C. Verified humans

**Input:** user $u$
**Output:** boolean
**if** $u.\mathrm{verified} \neq \mathrm{true}$ **then**
  **return** false
**end**
**if** $u.\mathrm{verified\_type} \in \{\mathrm{business}, \mathrm{none}\}$ **then**
  **return** false
**end**
**if** *handle or bio matches bot or API-generated patterns* **then**
  **return** false
**end**
**return** true

**Algorithm 2:** IsVerifiedHuman

### D. Budgeted parallel lookup

Let $B = 100$ be the official batch size and $W_{\max} = 8$ the worker cap [16], [17]. For a deduped handle list $H$,

$$N = \left\lceil |H|/B \right\rceil, \qquad b = \min(N, B_{\mathrm{req}}), \qquad w = \min(W, b, W_{\max}).$$

$B_{\mathrm{req}}$ is the pre-calculated request budget (`--budget` or `X_LOOKUP_BUDGET`). $W$ is `--workers` or `os.availableParallelism()`, capped at 8.

**Input:** handles $H$, budget $B_{\mathrm{req}}$, workers $W$
**Output:** users, optional block reason
plan $\leftarrow$ first $b$ batches of size $B$
**if** $b = 0$ **then**
  **return** empty
**end**
probe batch 1
**if** *HTTP* $402/429/401/403$ **then**
  **return** collected users and reason
**end**
shard remaining $b - 1$ batches into $w$ contiguous slices
**parallel for** each worker $i$
  **for** *batch in slice* $i$ **do**
    **if** *blocked* **then**
      **break**
    **end**
    lookup batch
    **if** *paywalled* **then**
      record reason and stop assigning
    **end**
**end**
**return** merged users

**Algorithm 3:** BudgetedLookup

The probe exists so a dead credit pool does not fire $w$ failing calls at once. Workers consume only their pre-assigned slice, so they cannot spend past $b$. Leftover handles stay seed placeholders.

### E. Handle-space walk

The optional enumerator appends one character from $\Sigma = \{a..z, 0..9, \_\}$ at a time [9]. It expands forward while the collected set is smaller than the handle target and the next layer fits the frontier. It walks backward when depth, target, or frontier is hit. Default depth is 2 and default lookup/handle/frontier caps are 2,000. `--depth` on `snapshot:live` does *not* start this walk; `--enumerate` does.

### F. Snapshot publish

Live humans overlay seed handles by case-insensitive username. Missing or unverified handles remain

TABLE II
LOCAL BENCHMARKS

| Workload | Result | Notes |
|---|---|---|
| Vitest suite | 118 pass / 1 skip | 402 ms wall |
| Typecheck | clean | `tsc --noEmit` |
| Seed snapshot write | 2,532–2,533 rows | no X API |
| Figure files | 24 | two directories |
| Seed unique handles | 2,533 | 1,212 / 1,321 |
| Needed lookup requests | 26 | $\lceil 2533/100 \rceil$ |
| Worker cap | 8 | after 1 probe |
| Parallel waves (full budget) | 4 | $\lceil 25/8 \rceil$ |
| Budget 5 overlay | 500 handles | rest stay seed |
| Handle space $\lvert\Sigma\rvert^{15}$ | $\sim 10^{23}$ | not enumerable |

`id=seed:<handle>` with tweet count 0. Celebrity placeholders receive $f = 10^6$ so they stay on the celebrity list. The worker writes JSON to `data/` and `web/data/` in parallel, then writes 24 figure files.

## IV. RUNTIME

### A. Sequential work

Deduping $|H|$ handles is $O(|H|)$. Sorting one list of $n$ seated rows is $O(n \log n)$ [12]. Human filtering is $O(n)$ regular-expression checks. Figure generation is linear in the drawn prefix.

### B. Network

One lookup request costs one credit unit in this model and covers up to $B = 100$ handles. Needed requests are $N = \lceil |H|/B \rceil$. Sequential latency is $N \cdot L$ where $L$ is per-request latency (connect timeout 30 s, request timeout 45 s). After the probe, remaining work runs in

$$\left\lceil \frac{b-1}{w} \right\rceil$$

waves. Amdahl's bound applies: the probe and any 402 stop are serial [18]. The public site never pays this cost.

### C. Handle space

$|\Sigma| = 37$, $\ell \le 15$, so $|\Sigma|^{15} \approx 8.4 \times 10^{23}$ strings. Even depth 2 is $37 + 37^2 = 1{,}406$ lookups if uncapped. The enumerator is a coverage pump, not a proof of global completeness.

### D. Memory

A fixture snapshot uses 1,000 seats per list so tests do not allocate 200,000 rows. The live/site cap remains 100,000 per list. The current seed is 2,533 unique handles, which is the observed snapshot size.

## V. BENCHMARKS

Measurements below are from the repository on 6 September 2026. They are local and deterministic except where marked.

The worker-pool unit test drives three batches of 100 and asserts peak in-flight concurrency 2 after the probe. The budget unit test asserts that 250 handles with `requestBudget`=2 fund two batches and leave 50 handles unspent.

A successful live overlay from an earlier session produced 98 live rows and a lifetime raw total of 3,269,095 tweets before credits ran out. A later `snapshot:live --depth unlimited` spent the pool, received HTTP 402, and—under a since-removed eligibility cut—wrote an empty board. That cut is gone. `npm run snapshot` restores the seed roster without touching the API.

We do not report a live latency table here. Any number collected after 402 would measure refusal, not lookup.

## VI. FUTURE WORK

- **Credit-aware planning.** Query remaining project credits and set $B_{\text{req}}$ from the vendor, not only from flags.
- **User-token search.** `users.search` is the official people sweep; it needs OAuth2 user context [8].
- **Lifetime engagement.** X does not expose lifetime impressions or likes on user lookup. Until it does, those columns stay zero.
- **Stronger bot models.** Pattern filters are transparent and brittle [13]. A reviewed classifier could sit behind the same port.
- **True global rank.** Impossible without a catalog. The handle walk cannot finish.
- **Worker threads.** The current pool is concurrent I/O on one event loop. CPU ranking of two full 100,000-row lists may later move to `worker_threads`.
- **Refresh cadence.** A scheduled seed overlay with a daily budget, instead of a manual live run.

## VII. CONCLUSION

Nanoblog Index ranks verified humans by raw lifetime tweet count on two 100,000-seat lists. The private worker is the only process allowed to call X. The public site reads a snapshot. Lookups are planned as $\lceil |H|/100 \rceil$ requests, probed once, then executed on at most eight workers inside a precalculated budget. HTTP 402 stops spend and falls back to seed placeholders. Frozen grades do not move. That is the entire ranking rule: tweet count, humans only, two lists, spend what you planned.

## REFERENCES


[1] X Corp., "Users lookup by username," X API v2 reference, `GET /2/users/by`, 2026, official user-object fields include `public_metrics.tweet_count`. [Online]. Available: https://docs.x.com/x-api/users/lookup

[2] E. Ferrara, O. Varol, C. Davis, F. Menczer, and A. Flammini, "The rise of social bots," *Communications of the ACM*, vol. 59, no. 7, pp. 96–104, 2016.

[3] Z. Chu, S. Gianvecchio, H. Wang, and S. Jajodia, "Detecting automation of Twitter accounts: Are you a human, bot, or cyborg?" in *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 6, 2012, pp. 811–824.

[4] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 3rd ed. Addison-Wesley, 2012.

[5] Vercel, "Next.js 15 documentation," 2025. [Online]. Available: https://nextjs.org/docs

[6] X Developers, "@xdevplatform/xdk," Official TypeScript SDK for the X API, 2026. [Online]. Available: https://www.npmjs.com/package/@xdevplatform/xdk

[7] D. Hardt, "The OAuth 2.0 authorization framework," RFC 6749, 2012.

[8] X Corp., "User search and recent post search," X API v2 reference, `GET /2/users/search` and `GET /2/tweets/search/recent`, 2026. [Online]. Available: https://docs.x.com/x-api
[9] ——, "Username rules," X Help Center, 2026, handles are 1–15 characters from $[A-Za-z0-9\_]$.
[10] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank citation ranking: Bringing order to the web," Stanford InfoLab, Tech. Rep., 1999.
[11] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of the ACM*, vol. 46, no. 5, pp. 604–632, 1999.
[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. MIT Press, 2009.
[13] S. Cresci, "A decade of social bot detection," *Communications of the ACM*, vol. 63, no. 10, pp. 72–83, 2020.
[14] R. Fielding, M. Nottingham, and J. Reschke, "HTTP semantics," RFC 9110, 2022, status 402 Payment Required.
[15] Node.js Undici, "Undici HTTP client," 2026. [Online]. Available: https://undici.nodejs.org
[16] Node.js Project, "`os.availableParallelism()`," Node.js documentation, 2026. [Online]. Available: https://nodejs.org/api/os.html
[17] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[18] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," *AFIPS Conference Proceedings*, vol. 30, pp. 483–485, 1967.