iota GATEWAY: ACTIVE EVIDENCE TRANSPORT FOR NEURAL WEIGHTS

RB LABS

Abstract. The **iota** gateway is a model access layer in which neural weights are represented as addressable evidence streams rather than only as completed files. A model artifact is split into a scale-bearing lane and progressively finer refinement lanes. A receiver may open the artifact from local disk, remote object storage, a network drive, or any range-readable host; fetch selected lanes; cache evidence locally; reconstruct a valid model state; and run inference without sending the user's prompt to the storage server. This document gives a ground-up formal specification of the **iota** gateway: floating point evidence, the Rouyea shave, lane decomposition, the crisp byte, prefix-stable topcut, zero-decode observability, evidence pricing, bit-packed and deflated lane storage, tensor-aware manifests, cached gateway retrieval, active evidence scheduling, native bounded-memory lane computation, and the Bourgeois orbience evaluator. The central claim is deliberately precise: model loading can be transformed from a binary file transfer into a continuous evidence negotiation, and model execution can be organized around bounded evidence streams rather than fully resident tensors.

Contents

Date: May 2026.

1. The Finding

A conventional model runner begins with a completed model file. It downloads or locates the full artifact, loads the full tensor state, and only then begins inference. The **iota** gateway changes the unit of access. A model may be opened before the whole model is downloaded. The first operation is not full loading; it is evidence acquisition.

> *The **iota** gateway makes neural models openable before they are downloaded, runnable before they are complete, and private because model evidence moves toward the user instead of user data moving toward the model.*

The thesis is

Model loading is evidence negotiation, not merely file transfer.

Equivalently, neural weights can be represented as a sequence of evidence states. Each state has a byte cost, an uncertainty, and an execution meaning.

1.1. The old loading model.

The usual local model flow is

$$\text{download model} \longrightarrow \text{load model} \longrightarrow \text{run inference}.$$

The usual cloud model flow is

$$\text{send prompt} \longrightarrow \text{remote inference} \longrightarrow \text{return tokens}.$$

The **iota** gateway flow is

$$\text{open manifest} \longrightarrow \text{fetch evidence} \longrightarrow \text{run locally} \longrightarrow \text{refine only when needed}.$$

The storage server serves bytes. It does not need the prompt. The local device or local gateway owns inference.

2. Floating Point as Evidence

A bfloat16 word has sixteen bits. Its high byte contains the sign and exponent-bearing structure; its low byte contains refinement bits. A standard runtime treats the word as an atomic number. The **iota** gateway treats it as a structured evidence object.

Definition 2.1 (Numeric profile)**.** A numeric profile is a quadruple $P = (w, s, e, m)$, where w is the word width in bits, s is the sign-bit count, e is the biased exponent width, and m is the explicit mantissa width, with $w = s + e + m$.

Definition 2.2 (bfloat16 profile)**.** The current **iota** profile is bfloat16, $P_{\text{bf16}} = (16, 1, 8, 7)$. Words are stored little-endian. The byte at offset $2i$ is the low byte. The byte at offset $2i + 1$ is the high byte.

Definition 2.3 (Evidence word)**.** An evidence word is a fixed-width floating-point word together with a prefix projection that exposes scale-bearing evidence first and detail-bearing evidence later.

3. The Rouyea Shave Algorithm

The primitive projection in **iota** is the Rouyea shave.

Definition 3.1 (Rouyea shave)**.** Let x be a bfloat16 word represented as an unsigned sixteen-bit integer. For $0 \le \kappa \le 8$, define

$$m_0 = \texttt{0xff00},$$

$$m_\kappa = \texttt{0xff00} \ \vee \ (\texttt{0xff} \ll 8 - \kappa) \quad \text{for } 1 \le \kappa \le 8.$$

The Rouyea shave at evidence depth κ is

$$\pi_\kappa(x) = x \ \& \ m_\kappa.$$

The operator extends elementwise to tensors.

Theorem 3.2 (Nested evidence)**.** *For $a \le b$,*

$$\pi_a(x) = \pi_a(\pi_b(x)).$$

Proof. The masks are nested: $m_a \& m_b = m_a$ when $a \le b$. Thus

$$\pi_a(\pi_b(x)) = (x \& m_b) \& m_a = x \& (m_b \& m_a) = x \& m_a = \pi_a(x).$$

□

Remark 3.3. The Rouyea shave is truncation, not rounding. Rounding can propagate a carry from discarded bits into retained bits and would break nested evidence.

4. Lane Decomposition

Definition 4.1 (Lane family)**.** For a sequence of bfloat16 words, the **iota** top decomposition produces one base lane and eight refinement lanes:

$$L_0 = \text{high byte stream},$$

$$L_j = \text{bit stream of low-byte bit } 8 - j, \quad 1 \le j \le 8.$$

A keep level κ uses lanes $L_0, L_1, \ldots, L_\kappa$.

The lane law is intentionally simple:

$$\text{keep } \kappa \quad \Longleftrightarrow \quad \text{use the first } 1 + \kappa \text{ lanes.}$$

This is why precision restriction can be expressed as prefix selection.

Definition 4.2 (Stitch)**.** Given lanes $(L_0, \ldots, L_\kappa)$, the stitch operator reconstructs a bfloat16 byte stream by placing $L_0[i]$ in byte $2i + 1$, setting retained low-byte bits from $L_1, \ldots, L_\kappa$, and setting omitted low-byte bits to zero.

Proposition 4.3 (Stitch equals shave)**.** *For any tensor T and keep κ, stitching the first $1 + \kappa$ lanes of the lane split equals $\pi_\kappa(T)$.*

Proof. Both procedures preserve the high byte, preserve the retained low-byte prefix bits, and zero all omitted low-byte suffix bits. □

5. The Crisp Byte

The artifact header contains a one-byte state field called the crisp byte.

Definition 5.1 (Crisp byte)**.** The crisp byte is a compact mode and keep descriptor. In the current top artifact profile, it contains:

bits	field	meaning
0	stable bit	artifact participates in the stable **iota** profile
1–4	keep	active keep level
5	top bit	progressive top-lane artifact
6	lattice bit	structural lattice mode
7	reserved	future use

The crisp byte is the state byte that lets a gateway know which evidence level an artifact declares.

6. Artifact Format

An **iota** top artifact is a finite byte string with the following canonical layout:

```
magic = "iota"
version
crisp
n_lanes
reserved
raw_size
lane_lengths[n_lanes]
payload_lanes
```

All multi-byte integers are little-endian. The lane table gives the byte length of every lane present in the artifact. Thus the gateway can locate every lane prefix without decoding model weights.

Definition 6.1 (Artifact)**.** A top-mode artifact at keep κ is

$$A_\kappa = H_\kappa \parallel \ell_0 \| \cdots \| \ell_\kappa \parallel E(L_0) \| \cdots \| E(L_\kappa),$$

where H_κ is the fixed header, $\ell_j = |E(L_j)|$ is the little-endian lane length, and E is the selected lane codec.

Definition 6.2 (Evidence cost)**.** For a lane L_j belonging to a tensor with N elements, define

$$\rho_j = \frac{\mathsf{bytes}(E(L_j))}{N}.$$

This is the lane density. It is the observed byte price of that evidence lane.

Proposition 6.3 (Header parseability)**.** *For any top artifact, the keep level, lane count, raw tensor size, lane lengths, and lane offsets are recoverable from the header and lane table without decoding any lane payload.*

Proof. These values are explicit fields or finite sums over explicit lane lengths. □

7. Lane Codecs

iota separates the semantic lane law from the physical codec. The lane law says which evidence appears in which order. The codec chooses how those lane bytes are stored.

Definition 7.1 (Strategy tag)**.** A strategy tag is a single byte prefixed to every encoded lane. The following tags are reserved:

tag	name	purpose
`0x00`	`Skip`	all-zero lane
`0x01`	`Rle`	run-length encoding
`0x02`	`Delta`	passthrough variant
`0x03`	`Raw`	uncompressed payload
`0x04`	`Fuse`	opcoded run and literal stream
`0x05`	`Lattice`	prototype and orbit fit
`0x06`	`Local`	drift alphabet with packed indices
`0x07`	`Drift`	zigzag byte-to-byte deltas
`0x08`	`BitPacked`	one bit per bit-plane entry
`0x09`	`Deflate`	conditional RFC1951 wrapper

Definition 7.2 (BitPacked lane)**.** For a bit-plane lane $L \in \{0,1\}^N$, `BitPacked` stores eight binary lane entries per byte, plus a finite length field sufficient to recover the exact original bit count. The decoder expands the packed stream back to the exact 0-or-1 lane used by stitch.

Proposition 7.3 (Bit-plane floor)**.** *For sufficiently large N, a bit-packed bit-plane has density approaching* $1/8$ *byte per element.*

Proof. The payload uses one bit per element. Byte alignment and finite strategy headers vanish in the large-N limit. □

Definition 7.4 (Deflate wrapper)**.** A deflate wrapper is a conditional lane codec. It is used only when the compressed representation is strictly smaller than the unwrapped lane representation.

Protocol rule 7.5 (Smallest winning lane)**.** For every lane, the encoder may test admissible exact codecs and keep the smallest byte representation that decodes to the required lane stream.

Remark 7.6. This rule has a useful consequence: structured exponent-bearing lanes may compress substantially, while low mantissa lanes may remain at the bit-packed noise floor when general compression cannot improve them.

8. Prefix-Stable Topcut

Definition 8.1 (Topcut)**.** Given an artifact containing lanes $L_0, \ldots, L_k$, the topcut operation to $q \leq k$ rewrites the header with keep q, sets the lane count to $1 + q$, writes the first $1 + q$ lane lengths, and copies exactly the corresponding payload prefix.

Theorem 8.2 (Prefix stability)**.** *For every source tensor W and keeps $q \leq k$, packing W directly at keep q produces the same lane payload as topcutting a keep k artifact to keep q, provided the same deterministic lane encoders are used.*

Proof. The Rouyea shave is nested and the lane family is ordered by prefix. Direct packing at keep q emits exactly lanes $L_0, \ldots, L_q$. Packing at keep k emits lanes $L_0, \ldots, L_q, \ldots, L_k$. Topcut copies the byte prefix containing $L_0, \ldots, L_q$ and rewrites only the header fields that declare the smaller keep. The retained payload bytes are identical. □

Corollary 8.3 (Zero-decode surgery)**.** *Topcut requires reading the header and lane-length table only. It performs no codec decode and no floating-point operation.*

9. Zero-decode Observability

Theorem 9.1 (Zero-decode observability)**.** *The keep level, raw size, lane count, lane lengths, lane offsets, prefix byte cost, and density of an **iota** artifact can be computed from the header and lane table without decoding any floating-point value.*

Proof. These quantities are explicit fields or finite sums over the lane-length table. No lane payload interpretation is required. □

This is the first gateway law. A remote storage object can be inspected by range-reading only the header and lane table.

10. Error Envelope

The current byte-native bfloat16 profile uses a conservative error envelope for missing low-byte evidence.

Definition 10.1 (Byte-native bfloat16 envelope)**.** Let $\epsilon(\kappa)$ denote a conservative relative unit envelope for the missing low-byte suffix. The profile used here is

$$\epsilon(0) = \frac{3}{4},$$
$$\epsilon(\kappa) = 2^{-(\kappa-1)} \quad \text{for } 1 \le \kappa \le 7,$$
$$\epsilon(8) = 0.$$

The envelope is intentionally conservative. Its purpose is safe refusal before useful calibration. Tighter residual radii belong to the bounded evaluator layer.

11. Tensor-Aware Artifact Folders

A whole-model artifact proves the lane law. A tensor-aware artifact makes the gateway sharp.

Definition 11.1 (Tensor-aware **iota** folder)**.** A tensor-aware **iota** folder is a model package containing a manifest and independent `.iota` artifacts for each tensor:

```
model.iota/
  manifest.json
  tensors/
    layer.0.attn.q_proj.weight.iota
    layer.0.attn.k_proj.weight.iota
    layer.0.mlp.down_proj.weight.iota
    ...
  config.json
  tokenizer.json
```

This layout permits heterogeneous evidence depth:

$$\kappa_{q\_proj} = 5, \quad \kappa_{k\_proj} = 2, \quad \kappa_{norm} = 8, \quad \kappa_{sleeping} = 0.$$

Uniform keep is a useful demo knob. Per-tensor keep is the gateway engine.

Definition 11.2 (Manifest)**.** A manifest is a machine-readable index containing at least: model identity, protocol version, tensor names, shapes, dtypes, element counts, lane sizes, densities, maximum keep, error envelope, and optional evaluator metadata.

Proposition 11.3 (Independent tensor addressability)**.** *In a tensor-aware folder, topcut applied to any tensor's artifact yields a valid keep-restricted artifact for that tensor, independent of every other tensor.*

Proof. Each tensor is stored as a self-contained artifact. Prefix stability applies per artifact. □

12. Storage Backend Interface

Definition 12.1 (Storage backend)**.** A storage backend provides three logical operations:

$$\mathsf{manifest} : \mathsf{ModelId} \to \mathsf{Manifest},$$

$$\mathsf{fetch\_range} : (\mathsf{ObjectId}, [a, b)) \to \mathsf{Bytes},$$

$$\mathsf{list\_models} : () \to [\mathsf{ModelId}].$$

A backend is generic when these operations can be implemented over ordinary byte-addressable storage, such as a local filesystem, HTTP Range, S3, R2, or a network drive.

Protocol rule 12.2 (HTTP Range conformance)**.** For a range request $[a, b)$, an HTTP backend issues a `GET` with `Range: bytes=`a`-`$b-1$, accepts a `206 Partial Content` response, and fails if the server ignores the range and returns `200 OK`.

13. The **iota** Gateway

Definition 13.1 (**iota** gateway)**.** An **iota** gateway is a local process that opens a model artifact from storage, reads its evidence map, fetches selected evidence lanes, caches evidence locally, reconstructs an executable model state, and runs or delegates inference without sending user prompts to the storage layer.

The storage layer may be local disk, external disk, a network share, static HTTP storage, S3, R2, or any host that supports byte ranges. The gateway does not require the storage layer to understand neural networks.

13.1. Gateway Roles.

role	object	responsibility
storage	bytes	serve range-readable artifacts
gateway	evidence router	read manifest, fetch lanes, cache, reconstruct
client	interface	prompt, inspect, select mode, display output
evaluator	runtime	execute reconstructed or native evidence state

Protocol rule 13.2 (Remote open)**.** To open a remote **iota** artifact, range-read the header and lane table first. The gateway may then compute the byte ranges required for any keep level before fetching the corresponding payload bytes.

Proposition 13.3 (Prompt locality)**.** *If storage is a static range server and inference is performed by the local consumer, then no prompt bytes are transmitted to the storage host.*

Proof. The storage host receives object identifiers and byte ranges. The prompt is consumed by the local evaluator after evidence retrieval. It is not an argument of the storage backend interface. □

Remark 13.4. This is structural privacy, not a logging policy. The storage service cannot log prompt content it never receives.

14. Caching

Definition 14.1 (Evidence cache)**.** An evidence cache stores fetched lanes or tensor-lane slices keyed by artifact identity, tensor identity, lane index, and byte range.

Definition 14.2 (Cached fetch)**.** A cached fetch returns the cached lane bytes when present; otherwise it delegates to the storage backend and stores the successful result.

Proposition 14.3 (Cache idempotence)**.** *For any repeated fetch of the same model, tensor, lane, and range, the storage backend is invoked at most once while the cache entry remains valid.*

15. Value of Evidence

Definition 15.1 (Allocation)**.** An allocation for a tensor-aware folder is a function

$$\alpha : \mathsf{TensorId} \to \{0, 1, \ldots, 8\}$$

assigning a keep level to each tensor. Its byte cost is

$$B(\alpha) = \sum_{t} \sum_{j=0}^{\alpha(t)} \mathsf{bytes}(E_t(L_j)).$$

Definition 15.2 (Value of evidence)**.** The value of evidence of a candidate tensor-lane is the expected reduction in uncertainty divided by the byte cost of acquiring it:

$$\mathsf{VoE}(t, j) = \frac{\mathbb{E}[\Delta U \mid t, j]}{\mathsf{bytes}(E_t(L_j))}.$$

Different applications choose different uncertainty functions U. A reconstruction test may use tensor error. A language-model gateway should eventually use orbit uncertainty, token entropy, margin, or a calibrated residual radius. The principle is stable:

load the evidence that changes the decision most per byte.

Definition 15.3 (VoE allocation)**.** Given sensitivity scores s_t and marginal improvements $r_{t,j}$, a greedy VoE scheduler promotes the tensor-lane maximizing

$$\frac{s_t r_{t,j}}{\mathsf{bytes}(E_t(L_j))}$$

until a budget or stopping condition is reached. Ties are resolved by a fixed deterministic order.

Proposition 15.4 (Scheduler determinism)**.** *For fixed sensitivities, costs, marginal improvements, budget, and tie-breaking order, the greedy VoE allocation is uniquely determined.*

16. The Bourgeois Orbience Algorithm

Definition 16.1 (Bourgeois orbience evaluator)**.** A Bourgeois orbience evaluator is a bounded evaluator

$$\mathcal{O}_\kappa(A, x) = (\ell_\kappa, R_\kappa, \Omega_\kappa).$$

Here A is an **iota** artifact or tensor-aware package, x is the runtime input, ℓ_κ is the partial orbit field, R_κ bounds possible movement caused by missing evidence, and Ω_κ denotes auxiliary geometry such as density, drift, mass, or sensitivity.

In language-model inference, the orbit field is the logit vector before softmax. Softmax normalizes the orbit field; sampling or argmax collapses it to an emitted token.

17. Certified Collapse

Let i_1 and i_2 index the two largest entries of ℓ_κ. Define the orbit gap

$$\Gamma_\kappa = \ell_\kappa[i_1] - \ell_\kappa[i_2].$$

Theorem 17.1 (Margin-aware certified collapse)**.** *If every logit can move by at most R_κ under all missing evidence, then the top token is stable whenever*

$$\Gamma_\kappa > 2R_\kappa.$$

Proof. In the worst case, the leading logit decreases by R_κ and the runner-up increases by R_κ. The leading logit remains larger if

$$\ell_\kappa[i_1] - R_\kappa > \ell_\kappa[i_2] + R_\kappa,$$

which is equivalent to $\Gamma_\kappa > 2R_\kappa$. □

This theorem is a safety rule. A loose radius may refuse to commit often. Tighter radii are the job of orbience calibration.

18. Active Gateway Scheduling

The active gateway loop is:

(1) Open the artifact manifest.
(2) Fetch a minimal evidence state.
(3) Evaluate the current orbit field.
(4) If collapse is certified, emit.
(5) Otherwise, select the next evidence packet by value of evidence.
(6) Fetch, cache, and repeat.

In a tensor-aware gateway, step five can select a lane from a specific tensor rather than increasing global keep for the whole model.

19. Native Bounded-Memory Execution

The gateway mechanism moves evidence. The native execution mechanism computes on evidence without requiring the full model to be resident as completed tensors.

Definition 19.1 (Lanview)**.** A lanview is a runtime view of one tensor artifact consisting of the decoded or directly gathered base lane, the active refinement state, the tensor shape, and the temporary buffers required for a matrix-vector operation. A lanview is bounded when its resident memory is proportional to the active tensor chunk and not to the whole model.

Definition 19.2 (Open)**.** The open operation maps an **iota** tensor artifact at keep κ into a lanview suitable for computation. A conforming open operation must preserve the value produced by stitch at keep κ. It need not materialize any representation not consumed by the evaluator.

Definition 19.3 (Direct bit-packed pregather)**.** For bit-packed mantissa lanes, direct pregather forms the gathered mantissa byte for each element directly from the packed lane bytes. For element index i, with byte index $q = \lfloor i/8 \rfloor$ and bit index $r = i \bmod 8$, the retained low-byte bits are read as

$$b_j(i) = (P_j[q] \gg r) \,\&\, 1,$$

where P_j is the packed payload for refinement lane j. The gathered low byte is

$$M(i) = \sum_{j=1}^{\kappa} b_j(i) 2^{8-j}.$$

This bypasses the intermediate expansion of each bit-plane into one byte per bit.

Proposition 19.4 (No expansion detour)**.** *Direct bit-packed pregather is value-equivalent to decoding every bit-packed lane into a* 0*-or-*1 *byte plane and then stitching, but it does not allocate the expanded bit-plane buffers.*

Proof. For every element i, both procedures read the same packed bit $b_j(i)$ for every active refinement lane j, place it in the same low-byte position $8 - j$, and preserve the same high byte. The direct procedure merely changes the order of operations: it composes bit extraction and low-byte gathering before allocation of any expanded lane plane. Thus the produced gathered low byte is identical, while the expanded intermediate buffers are absent. □

Definition 19.5 (Bounded-memory stream sweep)**.** A bounded-memory stream sweep over a model is an execution that visits model tensors sequentially or in bounded groups, opens each active tensor view, applies the required matrix-vector operation, accumulates the result, and releases tensor-local buffers before advancing. The peak resident footprint is controlled by the largest active working set rather than by the total model size.

Proposition 19.6 (Streaming memory law)**.** *Let a model contain tensors $T_1, \ldots, T_n$. Suppose an execution opens at most r tensor views at a time and each open view has working-set size at most W. If activation state and accumulator memory are bounded by A, then peak resident memory is bounded by*

$$A + rW + O(1),$$

independent of the total byte size $\sum_i |T_i|$, except through the chosen chunk or tensor working-set bound.

Proof. At each step, at most r tensor-local working sets are resident. Completed tensor buffers are released or reused before the next step. The remaining resident state is the activation and accumulator state plus fixed runtime overhead. The total model size determines the number of steps, not the peak resident memory. □

> *The bounded-memory rule is: active chunk size must fit in memory; total model size need not.*

Remark 19.7. This section describes weight execution, not the complete transformer graph. Attention, rotary embedding, tokenization, key-value cache policy, and graph-level scheduling are evaluator responsibilities. The gateway result is that weight access and matrix-vector computation can be organized as bounded evidence streams.

20. Conformance levels

Definition 20.1 (Conformance levels)**.** An implementation may declare any of the following levels:

L1 artifact reader: Parses headers, lane tables, topcut, stitch, and at least `Raw`, `Skip`, `BitPacked`, and `Deflate`.
L2 tensor-aware folder: Reads and writes tensor-aware manifests and per-tensor artifacts.
L3 range backend: Fetches lanes or lane prefixes over HTTP Range or an equivalent byte-addressable storage backend.
L4 gateway: Mediates between storage and a local evaluator while preserving prompt locality.
L5 cache: Implements evidence caching with idempotent repeated fetches.
L6 scheduler: Implements uniform allocation and a deterministic VoE allocation.
L7 bounded evaluator: Returns an orbit field and residual radius suitable for certified collapse.

21. Empirical Status

The following measurements summarize the current **iota** gateway and native bounded-compute validation on real Qwen-family artifacts and running Rust code. These numbers are implementation results, not axioms.

21.1. **Gateway Storage and Scheduling Results.**

quantity	previous state	current measured state
full lossless artifact	567% of raw bfloat16	70.3% of raw bfloat16
keep 4 fetch	310% of raw bfloat16	45.4% of raw bfloat16
keep 0 fetch	56% of raw bfloat16	20.6% of raw bfloat16
second prompt fetch	repeated remote fetch	0 remote bytes with cache
active global scheduling	not available	16% fewer bytes in measured run
tensor-aware allocation	not available	about 23% fewer bytes in measured run
per-tensor artifacts	not available	311 independent artifacts
lossless reassembly	not available	byte-identical model tensors
prompt privacy	policy claim	structural range-request property

These measurements establish the gateway mechanism: byte-identical reconstruction, lane correctness, range-fetch accounting, cache behavior, tensor-aware allocation, and prompt isolation. Token output demonstrates that the reconstructed model path executes through inference.

21.2. **Bounded-Memory Native Compute Results.** A later native stream sweep measured every two-dimensional weight matrix of an 8B-class model with raw size 15.26 GiB and 254 weight matrices. The sweep used one process, pure CPU execution, native **iota** compute, and the direct bit-packed pregather path described above.

keep	elapsed time	throughput	peak resident memory
0	7.3 s	1118 MB/s	2.46 GiB
4	13.4 s	914 MB/s	3.68 GiB
8	8.6 s	1896 MB/s	4.26 GiB

The keep 8 row is lossless. In this sweep, lossless matrix-vector evaluation over every measured weight matrix completed with peak resident memory equal to about 28% of the raw model size. The pure dot path after open measured approximately 154 GB/s; the measured bottleneck before refinement was not arithmetic but the open path.

21.3. **Decode-Wall Removal.** The runtime originally expanded bit-packed refinement lanes into one-byte 0-or-1 planes and then repacked those planes into gathered mantissa bytes. It also performed an unused base-lane orbience sample during shape encoding. Removing those detours changed the single-tensor open path and open-plus-dot cycle as follows:

operation	before	after
single-tensor open, 96 MiB tensor	about 3000 ms	about 47 ms
open plus dot cycle	about 3041 ms	about 32 ms
pure dot after open	about 149 GB/s	about 154 GB/s

The dot kernel did not require the major repair. The repair was to stop destroying the compact evidence form before computing on it. Thus the current implementation supports the following stronger interpretation: **iota** evidence is not only a storage representation; it is a compute-native representation.

21.4. **Interpretation.** These results validate a bounded-memory weight-execution claim, not a complete end-to-end transformer serving claim. They show that lossless weight computation can be organized around streamed evidence chunks rather than fully resident model tensors. Full graph integration, token-generation throughput, attention scheduling, key-value cache policy, and hardware-specific GPU kernels remain evaluator and runtime integration work.

22. Established Properties

The current implementation establishes the following:

- the Rouyea shave gives nested evidence states;
- lane decomposition supports prefix-stable precision restriction;
- the crisp byte and lane table make keep, size, density, and prefix cost observable without decoding floats;
- bit-packed mantissa lanes decode losslessly and reach the bit-plane storage floor;
- deflate improves structured lanes and declines noisy lanes when it does not win;
- remote static storage can serve **iota** evidence by byte range;
- a local gateway can reconstruct valid keep artifacts without sending prompts to storage;
- a cache can reduce repeated remote evidence fetches to zero bytes;
- tensor-aware artifacts permit heterogeneous per-tensor keep maps;
- per-tensor gateway fetch can reassemble byte-correct model tensors and run inference;
- direct bit-packed pregather avoids expanded bit-plane buffers while preserving stitch-equivalent values;
- a bounded-memory native stream sweep can execute lossless matrix-vector operations over all measured two-dimensional weights of an 8B-class model with peak resident memory below one-third of raw model size.

23. The Metric

The native **iota** metric is not merely compression ratio. It is

$$\frac{\text{bytes moved}}{\text{useful emitted token}}$$

or, when certified collapse is active,

$$\frac{\text{bytes moved}}{\text{certified emitted token}}.$$

This metric is meaningful because **iota** co-defines the evidence state, evidence cost, and collapse rule. A system that treats a model as simply loaded or unloaded cannot express the same quantity cleanly.

A companion systems metric is

$$\frac{\text{resident bytes}}{\text{lossless weight sweep}}$$

or, more operationally, peak resident memory as a fraction of raw model size for a lossless native sweep. This metric captures the bounded-memory distinction between model size and active working-set size.

24. Conclusion

The **iota** gateway makes neural weights behave less like static files and more like active evidence objects. The route is precise: the Rouyea shave creates ordered evidence; lanes make evidence addressable; the crisp byte declares evidence state; topcut restricts evidence by prefix; density prices evidence without decode; bit-packing and deflate make evidence efficient; the gateway fetches evidence from remote storage; the cache remembers it; tensor-aware manifests let each tensor carry its own keep; value-of-evidence selects what to acquire next; Bourgeois orbience evaluates partial evidence into an orbit field; and certified collapse determines when enough evidence has arrived.

The latest native execution results add the corresponding compute statement: a model is no longer only a tensor object that must be fully resident before useful lossless weight computation can occur. It can be a stream of executable evidence chunks.

The paradigm shift is small enough to implement and large enough to matter:

a model is no longer only a file to download; it is evidence to negotiate and compute.

References

[1] IEEE Computer Society. *IEEE Standard for Floating-Point Arithmetic*. IEEE Std 754-2019.

[2] L. P. Deutsch. *DEFLATE Compressed Data Format Specification version 1.3*. RFC 1951, 1996.

[3] R. Fielding et al. *Hypertext Transfer Protocol (HTTP/1.1): Range Requests*. RFC 7233, 2014.