Locality-Preserving Multiscale Computation for Compact Language Models

Space-filling curves, recursive regions, Q-Frontier selection, and evidence-preserving sparsity

Hayden Lindley · Glyphd Labs

2026-07-27

Contents

Locality-Preserving Multiscale Computation for Compact Language Models

Abstract

Transformer inference repeatedly moves and compares large amounts of state. Sparse attention, retrieval, caching, quantization, and small models reduce this burden, but their gains are often designed independently. The Glyphd fractal-computation program asks whether they can be coordinated through a shared multiscale geometry.

The proposal combines space-filling curves, recursive tilings, dynamic spatial indexes, foveated context selection, semantic object graphs, and a **Q-Frontier** that identifies the smallest unresolved set requiring expensive reasoning. Local detail is processed densely; increasingly distant or settled regions are represented more coarsely. Hilbert, Z-order, or Gosper-style orderings provide deterministic one-dimensional traversal of multidimensional neighborhoods. The geometry does not compute answers and does not compress arbitrary data. It organizes locality, candidate sparsity, cache layout, and evidence-preserving omission.

The program has two distinct levels. The first is practical runtime architecture around a compact model: structured outputs, PromptCapsules, prefix reuse, foveated retrieval, small deltas, and software scheduling. The second is model-architecture research: multiscale sparse attention and locality-preserving state layout. The first can be benchmarked now. The second requires kernels, trained models, and matched baselines before any performance claim.

1. The problem is movement as much as intelligence

A compact model can fail because it lacks learned capability. It can also fail because the runtime asks it to do unnecessary work:

- reread stable instructions;
- ingest the whole project;
- regenerate a whole page;
- inspect settled constraints;
- compare every token with every other token;
- decode long explanations when a small patch is enough;
- recompute cached prefixes;
- retrieve distant irrelevant history;
- repair malformed output.

The Max-Flex doctrine is to reduce the problem around the model.

```
small model
+ structured task
+ verified evidence
+ bounded neighborhood
+ stable cacheable prefix
+ deterministic renderer
+ verifier
```

This paper extends the doctrine into multiscale computation.

2. Locality as a shared primitive

Several systems need a definition of nearness:

- attention blocks;
- retrieval neighborhoods;
- semantic graph regions;
- cache segments;
- temporal episodes;
- visual surfaces;
- repeated motifs;
- dependency clusters.

If each system uses unrelated ordering, data is repeatedly rearranged and locality is lost between layers.

A shared locality representation can map multidimensional structure into deterministic ranges for:

- contiguous memory operations;
- prefetch;
- block-sparse attention;
- range retrieval;
- cache invalidation;
- progressive loading;
- distributed sharding.

Space-filling curves are candidate orderings because nearby points tend to remain near in the one-dimensional traversal.

3. Space-filling curves

Hilbert

Hilbert ordering has strong locality preservation for recursive square grids. It is useful for quadkey-like spatial data and fixed block traversals.

Z-order / Morton

Z-order interleaves coordinate bits. It is simpler and often cheaper to compute, with weaker locality in some cases.

Gosper / flowsnake

Gosper-style curves traverse recursive hexagonal tilings and may better fit six-neighbor or multi-directional topologies.

The architecture should not choose one curve by aesthetic preference. It should benchmark:

- locality metrics;
- block contiguity;
- kernel implementation cost;
- cache misses;
- routing overhead;
- sparsity quality;

- model task quality.

The curve is an index, not a proof of relevance.

4. Recursive tilings and aperiodic structure

Square grids are convenient, but information graphs may not naturally fit them. Candidate tilings include:

- quadtrees;
- octrees;
- hexagonal recursion;
- triangular subdivisions;
- aperiodic tilings;
- graph partitions.

Aperiodic or non-repeating structure may provide procedural landmarks and reduce visual repetition. It does not automatically improve compute.

The important abstraction is a nested region:

```
Region {
  region_id
  level
  members[]
  parent
  neighbors[]
  summary_ref
  unresolved_frontier[]
  cache_ref?
}
```

Regions can be defined over tokens, semantic nodes, events, source chunks, or model states.

5. Dense local, sparse distant

The core attention hypothesis is:

```
dense attention within the active local region
+ selected cross-region edges
+ progressively coarser distant summaries
```

A token or semantic node may attend to:

- immediate neighbors;
- enclosing region summary;
- sibling summaries;
- selected evidence anchors;
- unresolved constraints;
- long-range bridge nodes.

This resembles many sparse and hierarchical attention ideas. The Glyphd-specific contribution is the composition with canonical evidence, FOVEA retrieval, SurfaceDelta output, and an explicit unresolved frontier.

No claim is made that one geometric mask is universally optimal.

6. Q-Frontier

The **Q-Frontier** is the set of objects whose resolution can materially change the answer or action.

A frontier candidate may have:

- unresolved contradiction;
- high authority;
- missing evidence;
- high downstream dependency;
- high uncertainty;
- action risk;
- time pressure;
- boundary position between semantic regions;
- repeated failure;
- stale state.

A software priority can be:

```
priority ∝
    authority criticality
  + uncertainty
  + action risk
  + evidence need
  + dependency count
  + time sensitivity
  - verified stability
  - recent redundant compute
```

Objects outside the frontier can be:

- returned from exact cache;
- represented by summaries;
- omitted with an omission record;
- loaded only if the frontier expands.

The frontier is not the model's hidden chain of thought. It is a public operational selection of which state deserves compute.

7. Evidence-preserving sparsity

Ordinary pruning risks dropping the source that would falsify an answer. A Glyphd sparsity mask retains typed anchors:

- authorizing constraints;
- blocking constraints;
- exact evidence;
- unresolved contradictions;
- source spans;
- dependency roots;
- current objective;
- accepted state;
- explicit unknowns.

The context compiler emits an omission manifest:

```
included regions
excluded regions
```

```
selection reason
proof anchors retained
budget
frontier state
expansion route
```

A result can therefore state not only what it used but what it intentionally did not load.

8. Runtime-level multiscale stack

Before changing model architecture, the same ideas can be implemented around an existing model.

Exact and semantic stores

Content-addressed sources and semantic graph nodes.

FOVEA retrieval

Active object, parents, siblings, exact evidence, recent mutation, landmarks.

PromptCapsule

Stable system/schema prefix plus bounded evidence tail.

Structured decoding

One of a small set of output schemas.

SurfaceDelta

Small patch rather than full answer.

Prefix and KV reuse

Model-specific caches keyed by exact weights, tokenizer, runtime, position profile, and source spans.

Software TTU

Route deterministic work, local model work, specialist model work, and stronger verification according to risk.

This stack is immediately testable without training a new transformer.

9. Architecture-level research

A future model may internalize region structure.

Candidate design:

```
token embeddings
→ locality map
→ local dense blocks
→ region summaries
```

```
→ bridge-node attention
→ Q-Frontier refinement
→ structured output head
```

Training would need:

- standard language tasks;
- long-context tasks;
- retrieval;
- source-grounded reasoning;
- structured state mutation;
- ablation of curve/tiling choices;
- fixed compute comparisons.

The model must be compared at equal parameter count, training tokens, context, hardware, and wall-clock budget.

10. Dynamic spatial indexes

Static tilings can become unbalanced. Dynamic structures may include:

- quadtrees with occupancy thresholds;
- cover trees;
- HNSW-like semantic neighborhoods;
- interval trees;
- scene graphs;
- dependency DAG partitions.

Dynamic rebalancing must not change logical identity or canonical addresses. It can create a new compute projection or representation.

A project can preserve stable FOVEA place while the model's internal compute partition changes.

11. Cache locality

A locality-preserving order may improve:

- contiguous sparse blocks;
- reduced gather/scatter;
- prefetch;
- page locality;
- sharding;
- prefix segment reuse;
- invalidation of local SurfaceDeltas.

It may also add curve conversion and metadata cost. GPU kernels may already favor simple block layouts. Measurements must include total pipeline time, not only theoretical neighbor distance.

12. Fractal computation and “compression”

The safe claims are:

- recursive structure can organize multiscale context;
- space-filling curves can provide deterministic locality-preserving traversal;
- sparse masks can reduce candidate comparisons;
- procedural geometry can generate landmarks;

- repeated motifs can support dictionary/macro reuse;
- foveated retrieval can reduce data movement.

The banned claim is:

> fractals themselves compress arbitrary data or compute answers.

Any byte saving belongs to a codec, dictionary, deduplication, cache, or omission policy and must be measured.

13. Current evidence

The project corpus contains:

- a detailed conceptual framework for Hilbert, Z-order, Gosper, tilings, and dynamic indexes;
- FOVEA address and Hilbert mechanisms;
- Max-Flex runtime contracts;
- PromptCapsules and SurfaceDelta;
- GlyphCAS semantic and exact DAGs;
- Q-Frontier and software TTU concepts;
- local model probability and tensor capture artifacts;
- MicroScope causal-search work over local model internals.

The evidence supports a coherent research program and some runtime components. It does not prove a trained fractal transformer or GPU speedup.

14. Benchmark ladder

Level 1 — ordering

Synthetic and real graph locality:

- neighbor distance;
- range fragmentation;
- cache lines/pages;
- conversion cost;
- update cost.

Level 2 — runtime retrieval

- tokens moved;
- source retention;
- task score;
- latency;
- cache hit;
- omission error;
- frontier expansion count.

Level 3 — sparse attention kernels

- throughput;
- memory;
- prefill;
- decode;
- kernel occupancy;
- model quality;

- curve/tiling ablations.

Level 4 — trained compact model

- same parameter count;
- same training data and tokens;
- same hardware;
- long-context quality;
- retrieval;
- structured output;
- energy;
- robustness.

Level 5 — end-to-end product

- time to verified useful result;
- user corrections;
- citation;
- project continuity;
- device thermals;
- cost.

15. Kill criteria

- curve ordering gives no useful locality after conversion overhead;
- sparse masks lose critical evidence;
- Q-Frontier ranking fails to prioritize answer-changing state;
- dynamic partitions create unacceptable churn;
- model quality falls more than compute improves;
- cache gains disappear under realistic invalidation;
- runtime stack adds more latency than it saves;
- a simpler baseline achieves the same result.

16. Security

Sparse context can become a security vulnerability if it omits policy or blocking state. The compiler must retain mandatory anchors independent of learned relevance.

Threats include:

- prompt injection manipulating frontier priority;
- hidden authority;
- cross-project neighbor leakage;
- cache poisoning;
- stale region summaries;
- malicious macro expansion;
- model-specific cache misuse.

Exact keys, project scope, authority, dependency versions, and verifier gates are required.

17. Limitations

Geometric locality may not match semantic or causal relevance. Space-filling curves can preserve proximity imperfectly. Attention patterns are task-dependent. Dynamic graphs can

be expensive. Sparse kernels are difficult to implement efficiently. Aperiodic tilings may be visually or mathematically interesting without practical value.

The program spans interface, runtime, and model architecture. Evidence from one layer cannot be promoted to another.

18. Conclusion

The fractal-computation program is not a claim that language is a fractal. It is a maximum-seeking attempt to make locality, scale, and unresolved state first-class throughout the stack.

The practical first move is modest:

> Give the model the smallest verified neighborhood and ask it for the smallest valid delta.

The longer research move is ambitious:

> Coordinate retrieval, cache, sparse attention, and structured output through multi-scale locality—and kill every geometric component that fails a matched baseline.

Architecture illustration briefs

- **P13-F01 — Multiscale compute manifold**: Tokens or semantic nodes placed in nested regions, traversed by Hilbert/Gosper paths, with dense local and sparse bridge attention.
- **P13-F02 — Q-Frontier**: Project graph with verified settled regions faded and unresolved high-authority/evidence nodes highlighted as the compute frontier.
- **P13-F03 — Runtime gains stack**: Exact store → FOVEA → PromptCapsule → structured decode → SurfaceDelta → verifier → renderer, with measured quantities.
- **P13-F04 — Evidence-preserving sparsity mask**: Sparse attention/retrieval neighborhood retaining mandatory source, authority, blocking, and contradiction anchors.

Source register

- **P13-S01** — Fractal Frameworks for LLMs project conversation (2026-07-24). `CUTTING EDGE OR NEW / Fractal Frameworks for LLMs`
- **P13-S02** — Zeke 4B Max-Flex Runtime Master Spec (2026-07-15). `ZEKE-4B-MAXFLEX-RUNTIME-MASTER-SPEC-v0.1.md`
- **P13-S03** — GlyphDrive FOVEA Specification (2026-07-12). `GLYPHDRIVE-FOVEA-SPEC-v0.2.md`
- **P13-S04** — GlyphCAS Design Report (2026-07-24). `How_we_can_make_CAS_lightning_fast_for_any_LL`
- **P13-S05** — MicroScope Resolution Frontier checkpoint (2026-03-27). `PHASE5_CHECKPOINT_SUMMARY.md`

Public-disclosure and IP boundary

This paper publishes the public-safe research architecture, boundaries, benchmarks, and kill criteria. It excludes unpublished model designs, kernel implementations, private training plans, patent-claim language, and confidential experiment data. No trained fractal transformer or performance advantage is claimed.