DeepSeek in the Broader Lineage

Deep Dive into DeepSeek — Appendix B

Yin Yang

Foundation Books Project

The one rule of this appendix

"Invented here" versus "borrowed" is too coarse
to explain how DeepSeek assembles established ideas.

- DeepSeek's best-known mechanisms belong to *different* histories: attention caching, conditional computation, reinforcement learning, sparse attention, low-precision formats.
- Each history needs its own evidence. MLA needs cached representations; MoE needs routing *and* transport; R1 needs a recipe; DSA needs a selector; low precision needs an executable numeric path.
- So the job is to locate the DeepSeek-specific change inside each history, and to follow only the consequences that public artifacts expose.

Goal: one attribution that distinguishes broad problem lineage, DeepSeek-specific realization, and released system surface — and that stops exactly where the public record stops.

Roadmap

1. Five mechanisms, five histories

Roadmap

1. Five mechanisms, five histories
2. One pinned decode token

Roadmap

1. Five mechanisms, five histories
2. One pinned decode token
3. Cache and routing

Roadmap

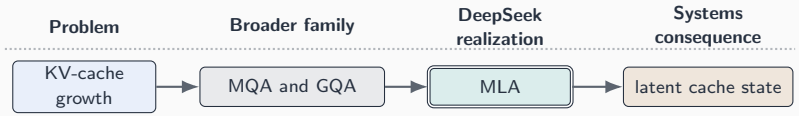
1. Five mechanisms, five histories
2. One pinned decode token
3. Cache and routing
4. Recipe, selector, and number

Roadmap

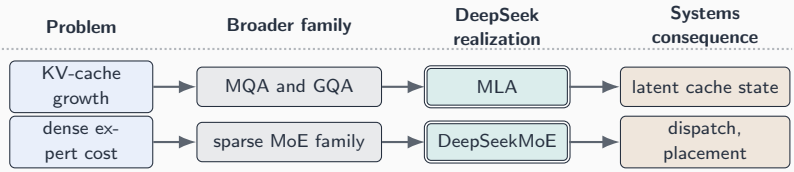
1. Five mechanisms, five histories
2. One pinned decode token
3. Cache and routing
4. Recipe, selector, and number
5. From concept to code

Five mechanisms, five histories

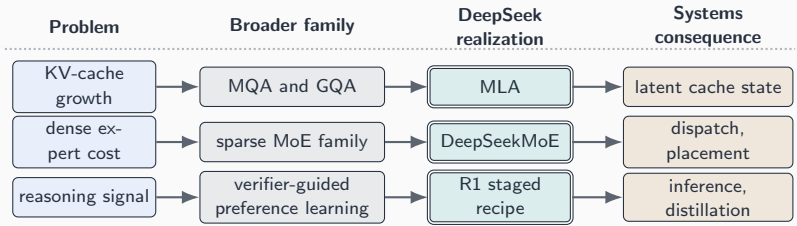
The five lineage rows, at a glance



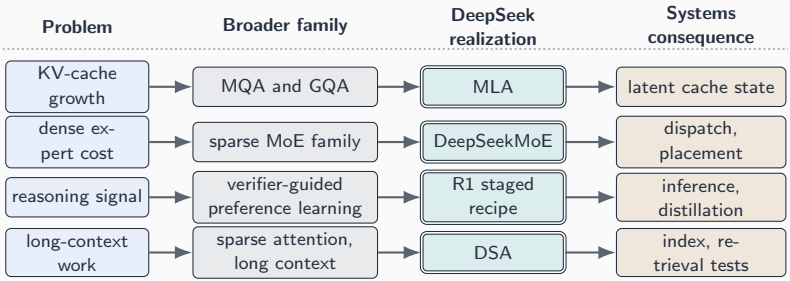
The five lineage rows, at a glance



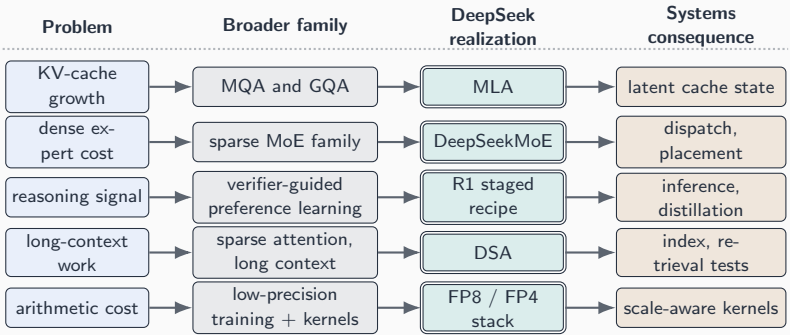
The five lineage rows, at a glance



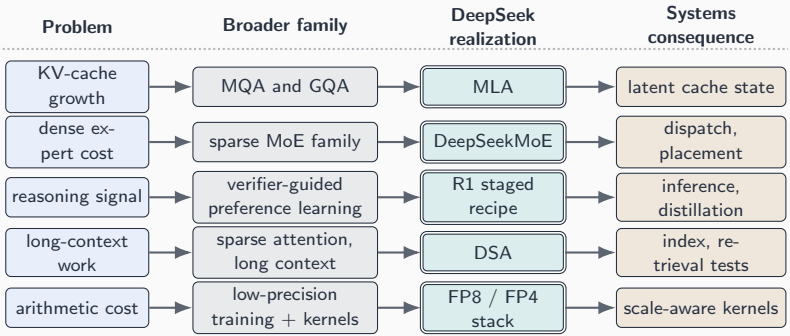
The five lineage rows, at a glance



The five lineage rows, at a glance



The five lineage rows, at a glance



Double-bordered boxes are the DeepSeek realization. The arrows encode *lineage and adaptation* — not invention priority and not a common chronology.

Three edge types, not one attribution formula

Edge type	What it establishes	What it cannot establish
Broad problem lineage	the older family a mechanism answers	that DeepSeek originated the primitive

Three edge types, not one attribution formula

Edge type	What it establishes	What it cannot establish
Broad problem lineage	the older family a mechanism answers	that DeepSeek originated the primitive
DeepSeek realization	the specific change in the public artifacts	that every idea in the family is DeepSeek's

Three edge types, not one attribution formula

Edge type	What it establishes	What it cannot establish
Broad problem lineage	the older family a mechanism answers	that DeepSeek originated the primitive
DeepSeek realization	the specific change in the public artifacts	that every idea in the family is DeepSeek's
Released system surface	what a public kernel or library exposes	that the surfaces were wired into one production engine

- Background citations carry the broad lineage; DeepSeek artifacts carry the DeepSeek-specific claim. Neither citation type proves the other.
- Compatibility among public artifacts is not deployment evidence.

Three edge types, not one attribution formula

Edge type	What it establishes	What it cannot establish
Broad problem lineage	the older family a mechanism answers	that DeepSeek originated the primitive
DeepSeek realization	the specific change in the public artifacts	that every idea in the family is DeepSeek's
Released system surface	what a public kernel or library exposes	that the surfaces were wired into one production engine

- Background citations carry the broad lineage; DeepSeek artifacts carry the DeepSeek-specific claim. Neither citation type proves the other.
- Compatibility among public artifacts is not deployment evidence.

The five sections use the evidence native to each mechanism instead of repeating the four columns as a prose template.

One pinned decode token

The running example, pinned to a public configuration

Quantity	Value	Role
main decoder layers	61	cache ledger multiplier
attention heads	128	MHA counterfactual width
key width	$192 = 128 + 64$	non-RoPE + RoPE
value width	128	MHA counterfactual width
latent KV rank	512	MLA retained state
positional-cache width	64	MLA retained state
routed experts / selected	256 / 8	sparse-compute fraction
shared-expert equivalent	1	always-active path

One retained BF16 decode position, one MoE layer. This is an accounting trace, not an observed production request. [REPORTED: DeepSeek-V3 repository and technical report]

Cache and sparse-compute attribution for that token

A materialized MHA counterfactual would retain

$$w_{\mathrm{MHA}} = 128\,(192 + 128) = 40{,}960$$

key/value scalars per token per layer.

Cache and sparse-compute attribution for that token

A materialized MHA counterfactual would retain

$$w_{\mathrm{MHA}} = 128\,(192 + 128) = 40{,}960$$

key/value scalars per token per layer.

The absorbed V3 MLA account instead retains

$$w_{\mathrm{MLA}} = 512 + 64 = 576.$$

Cache and sparse-compute attribution for that token

A materialized MHA counterfactual would retain

$$w_{\mathrm{MHA}} = 128\,(192 + 128) = 40{,}960$$

key/value scalars per token per layer.

The absorbed V3 MLA account instead retains

$$w_{\mathrm{MLA}} = 512 + 64 = 576.$$

So the ratio is $\dfrac{w_{\mathrm{MHA}}}{w_{\mathrm{MLA}}} \approx 71.1$.

Cache and sparse-compute attribution for that token

A materialized MHA counterfactual would retain

$$w_{\mathrm{MHA}} = 128\,(192 + 128) = 40{,}960$$

key/value scalars per token per layer.

The absorbed V3 MLA account instead retains

$$w_{\mathrm{MLA}} = 512 + 64 = 576.$$

So the ratio is $\frac{w_{\mathrm{MHA}}}{w_{\mathrm{MLA}}} \approx 71.1$. Across all 61 layers, in BF16, that one retained position costs $61 \cdot 576 \cdot 2 = 70{,}272$ bytes.

Cache and sparse-compute attribution for that token

A materialized MHA counterfactual would retain

$$w_{\mathrm{MHA}} = 128\,(192 + 128) = 40{,}960$$

key/value scalars per token per layer.

The absorbed V3 MLA account instead retains

$$w_{\mathrm{MLA}} = 512 + 64 = 576.$$

So the ratio is $\frac{w_{\mathrm{MHA}}}{w_{\mathrm{MLA}}} \approx 71.1$. Across all 61 layers, in BF16, that one retained position costs $61 \cdot 576 \cdot 2 = 70{,}272$ bytes.

At one MoE layer the token also selects $8/256 = 3.125\%$ of the routed-expert pool and traverses the shared path.

Cache and sparse-compute attribution for that token

A materialized MHA counterfactual would retain

$$w_{\mathrm{MHA}} = 128\,(192 + 128) = 40{,}960$$

key/value scalars per token per layer.

The absorbed V3 MLA account instead retains

$$w_{\mathrm{MLA}} = 512 + 64 = 576.$$

So the ratio is $\dfrac{w_{\mathrm{MHA}}}{w_{\mathrm{MLA}}} \approx 71.1$. Across all 61 layers, in BF16, that one retained position costs $61 \cdot 576 \cdot 2 = 70{,}272$ bytes.

At one MoE layer the token also selects $8/256 = 3.125\%$ of the routed-expert pool and traverses the shared path.

That fraction describes selected routed *functions*. It is not the model's active-parameter ratio, and it excludes attention and shared-expert work. [DERIVED: public V3 configuration]

What the pinned token deliberately excludes

- **FP8.** The V3 report supports an FP8 training framework — but the BF16 cache ledger above does not thereby become an FP8 decode result. An FP8 or FP4 execution claim additionally needs the payload format, scale layout, consuming kernel, accumulator, and tolerance.

What the pinned token deliberately excludes

- **FP8.** The V3 report supports an FP8 training framework — but the BF16 cache ledger above does not thereby become an FP8 decode result. An FP8 or FP4 execution claim additionally needs the payload format, scale layout, consuming kernel, accumulator, and tolerance.
- **R1.** Excluded: it is a training and distillation recipe, not a state transition inside this V3 decode token.

What the pinned token deliberately excludes

- **FP8.** The V3 report supports an FP8 training framework — but the BF16 cache ledger above does not thereby become an FP8 decode result. An FP8 or FP4 execution claim additionally needs the payload format, scale layout, consuming kernel, accumulator, and tolerance.
- **R1.** Excluded: it is a training and distillation recipe, not a state transition inside this V3 decode token.
- **DSA.** Excluded: introduced on the later V3.2 sparse long-context path, not the pinned V3 record.

What the pinned token deliberately excludes

- **FP8.** The V3 report supports an FP8 training framework — but the BF16 cache ledger above does not thereby become an FP8 decode result. An FP8 or FP4 execution claim additionally needs the payload format, scale layout, consuming kernel, accumulator, and tolerance.
- **R1.** Excluded: it is a training and distillation recipe, not a state transition inside this V3 decode token.
- **DSA.** Excluded: introduced on the later V3.2 sparse long-context path, not the pinned V3 record.

Warning (Exclusion is part of the attribution)
Naming what a record cannot carry is what keeps the three edge types distinct. A completed attribution says where it stops.

Cache and routing

MLA inside cache-efficient attention

Question	Broader lineage answer	DeepSeek-specific answer
What is the pressure?	decode repeatedly reads per-position attention state	MLA is reported as a way to reduce the V2/V3 KV-cache bottleneck
What changes?	MQA and GQA change key/value head sharing	MLA stores a low-rank latent KV representation plus position-sensitive state
What is distinctive?	cache sharing is a broad serving trend	the DeepSeek line makes cache *representation* a family signature and a kernel story
What must not be inferred?	MQA/GQA background does not specify MLA	MLA sources do not prove all cache compression is DeepSeek's

Same pressure, two different stored objects: share ordinary heads, or compress the state.
FlashMLA then realizes the compressed representation in kernels and cache interfaces.

MoE: overclaim, bounded claim, evidence

Overclaim	Bounded claim	Supporting evidence
DeepSeek invented MoE routing.	DeepSeek uses and extends sparse expert routing inside an older lineage.	earlier MoE literature for the family; DeepSeekMoE, V2, V3 for DeepSeek claims
Active parameters explain everything.	dispatch, balance, communication, and combine are also cost columns.	DeepEP and EPLB, only for the surfaces they expose
Routing is just a top-k formula.	a working system also needs route records, capacity rules, packing, reconstruction invariants.	DeepSeekMoE/V2/V3 for semantics; DeepEP/EPLB for transport

DeepSeek's credit becomes precise at the complete systems path, not at the generic idea of conditional computation.

Recipe, selector, and number

R1: four separate attribution objects

Object	Broader lineage	DeepSeek-specific role
Reward signal	human preference, learned reward models, rule rewards, verifier feedback	R1-Zero emphasizes rule-verifiable reasoning rewards in a public pure-RL story
Policy update	PPO-family and preference-optimization traditions, RLHF systems frameworks	GRPO becomes the named DeepSeek update mechanism
Data construction	rejection sampling, filtering, SFT, distillation	a staged recipe: cold start, reasoning RL, sampled data, mixed SFT, final RL, distillation
Released artifact	serving routes, model cards, open-weight distilled learners	DeepSeek makes the recipe legible through releases

"Just RLHF" misses the structure. A source for one ingredient cannot establish the whole sequence.

DSA: the selector is named, the measurements are not settled

- The general problem is quadratic attention *and* long-context evidence recall. Accepting more tokens does not guarantee that the model recalls the evidence it needs.
- Native Sparse Attention supplies mechanism background for broad, fine-selected, and local sparse paths.
- V3.2-Exp introduces DeepSeek Sparse Attention for long-context efficiency; the V3.2 paper ties DSA to a lightning indexer, fine-grained top-k token selection, and MLA under an MQA-style mode. [REPORTED: DeepSeek-V3.2 release and paper]
- The V4 report continues the sparse/compressed attention line in the next model generation. [REPORTED: DeepSeek-V4 report]

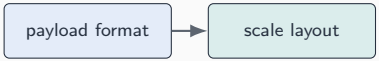
Naming the selector does not settle the two measurements that motivate sparse attention: how much work it saves, and how often it preserves the evidence the task needs.

FP8 and FP4: four parts of one numerical claim

payload format

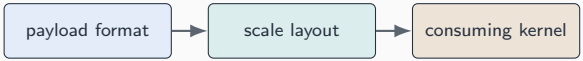
- A payload name by itself is never the whole numerical claim. FP4 alone does not establish lower serving cost.
- V3 reports an FP8 mixed-precision training framework; FlashMLA and DeepGEMM expose low-precision cache, attention, GEMM, and MoE paths; the V4 report adds FP4 QAT and FP4/FP8 paths. [REPORTED: V3 and V4 reports; kernel repositories]
- The practical test: which tensor uses the format, how is it scaled, which kernel consumes it, and what tolerance or quality result makes that path acceptable?

FP8 and FP4: four parts of one numerical claim



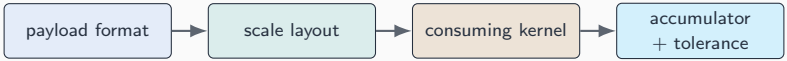
- A payload name by itself is never the whole numerical claim. FP4 alone does not establish lower serving cost.
- V3 reports an FP8 mixed-precision training framework; FlashMLA and DeepGEMM expose low-precision cache, attention, GEMM, and MoE paths; the V4 report adds FP4 QAT and FP4/FP8 paths. [REPORTED: V3 and V4 reports; kernel repositories]
- The practical test: which tensor uses the format, how is it scaled, which kernel consumes it, and what tolerance or quality result makes that path acceptable?

FP8 and FP4: four parts of one numerical claim



- A payload name by itself is never the whole numerical claim. FP4 alone does not establish lower serving cost.
- V3 reports an FP8 mixed-precision training framework; FlashMLA and DeepGEMM expose low-precision cache, attention, GEMM, and MoE paths; the V4 report adds FP4 QAT and FP4/FP8 paths. [REPORTED: V3 and V4 reports; kernel repositories]
- The practical test: which tensor uses the format, how is it scaled, which kernel consumes it, and what tolerance or quality result makes that path acceptable?

FP8 and FP4: four parts of one numerical claim



- A payload name by itself is never the whole numerical claim. FP4 alone does not establish lower serving cost.
- V3 reports an FP8 mixed-precision training framework; FlashMLA and DeepGEMM expose low-precision cache, attention, GEMM, and MoE paths; the V4 report adds FP4 QAT and FP4/FP8 paths. [REPORTED: V3 and V4 reports; kernel repositories]
- The practical test: which tensor uses the format, how is it scaled, which kernel consumes it, and what tolerance or quality result makes that path acceptable?

From concept to code

From concept to code: the cache counterfactual is in the repo

The MHA-versus-MLA comparison in the pinned example is not a thought experiment — the reference implementation registers both.

from `DeepSeek-V3/inference/model.py`

```
if attn_impl == "naive":
    self.register_buffer("k_cache", torch.zeros(..., self.n_local_heads, self.qk_head_dim), ...)
    self.register_buffer("v_cache", torch.zeros(..., self.n_local_heads, self.v_head_dim), ...)
else:
    self.register_buffer("kv_cache", torch.zeros(..., self.kv_lora_rank), ...)
    self.register_buffer("pe_cache", torch.zeros(..., self.qk_rope_head_dim), ...)
```

From concept to code: the cache counterfactual is in the repo

The MHA-versus-MLA comparison in the pinned example is not a thought experiment — the reference implementation registers both.

from `DeepSeek-V3/inference/model.py`

```
if attn_impl == "naive":
    self.register_buffer("k_cache", torch.zeros(..., self.n_local_heads, self.qk_head_dim), ...)
    self.register_buffer("v_cache", torch.zeros(..., self.n_local_heads, self.v_head_dim), ...)
else:
    self.register_buffer("kv_cache", torch.zeros(..., self.kv_lora_rank), ...)
    self.register_buffer("pe_cache", torch.zeros(..., self.qk_rope_head_dim), ...)
```

The `naive` branch carries the head count; the absorbed branch carries `kv_lora_rank` and `qk_rope_head_dim` instead — the 40,960 versus 576 split, written as tensor shapes. [INSPECTED: DeepSeek-V3 `inference/model.py`]

Concept-to-code map for the appendix

- **latent cache state vs. head-shaped cache** → `DeepSeek-V3/inference/model.py`
- **route weights, expert ids, shared path** → `DeepSeek-V3/inference/model.py`
- **lightning indexer and top-k budget** → `DeepSeek-V3.2-Exp/inference/model.py`
- **kernel surface for MLA cache state** → `FlashMLA/README.md`
- **low-precision GEMM and MoE paths** → `DeepGEMM/README.md`
- **dispatch, combine, and placement** → `DeepEP/README.md`, `EPLB/README.md`
- **staged reasoning recipe and distilled learners** → `DeepSeek-R1/README.md`

An inherited mechanism family, reorganized around a concrete architecture, training sequence, or systems realization.

Next: the appendix's five sections, each following the evidence native to its mechanism — and each stopping where the public record stops.