Scaling the Memory Wall

Towards 3D-DRAM-Based Accelerators
for Efficient Generative Inference

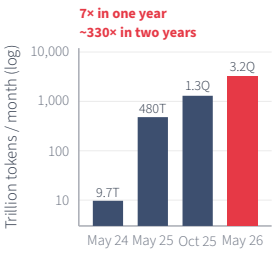
Prashant J. Nair

External Talk · IBM

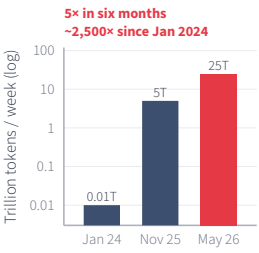
2026: The Year of Agents

Token consumption is compounding — and speed is a priced product

Google: tokens per month



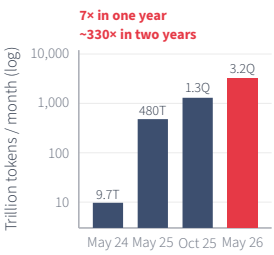
OpenRouter: tokens per week



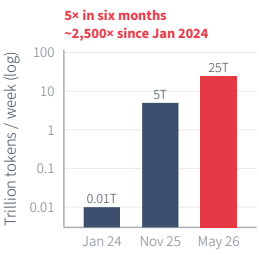
Sources: Google I/O 2026 keynote and Oct. 2025 Cloud event (via press coverage); OpenRouter figures as reported by I/O Fund and others, May 2026 (totals are not agent-specific); third-party pricing trackers (Finout, Sentra, claudefa.st, eesel) citing Anthropic and OpenAI list prices, Sept.–Oct. 2026.

Token consumption is compounding — and speed is a priced product

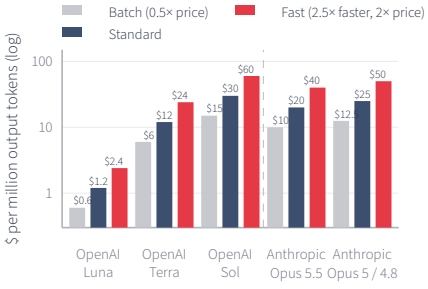
Google: tokens per month



OpenRouter: tokens per week



List price by service tier

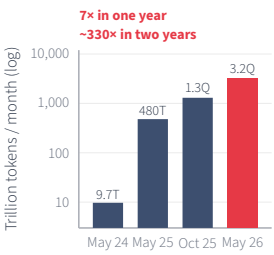


Sources: Google I/O 2026 keynote and Oct. 2025 Cloud event (via press coverage); OpenRouter figures as reported by I/O Fund and others, May 2026 (totals are not agent-specific); third-party pricing trackers (Finout, Sentra, claudefa.st, eesel) citing Anthropic and OpenAI list prices, Sept.–Oct. 2026.

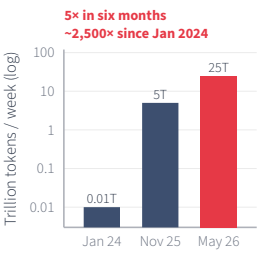
2026: The Year of Agents

Token consumption is compounding — and speed is a priced product

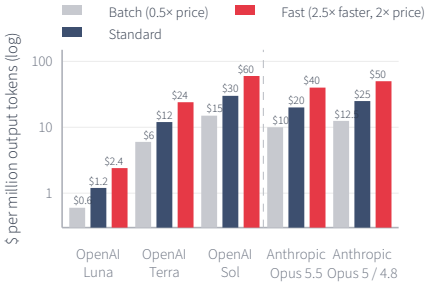
Google: tokens per month



OpenRouter: tokens per week



List price by service tier



Sources: Google I/O 2026 keynote and Oct. 2025 Cloud event (via press coverage); OpenRouter figures as reported by I/O Fund and others, May 2026 (totals are not agent-specific); third-party pricing trackers (Finout, Sentra, claudefa.st, eesel) citing Anthropic and OpenAI list prices, Sept.–Oct. 2026.

Agentic workloads multiply token consumption,
and **low-latency tokens carry a price premium**

Generative Inference Has Two Phases

Prefill

Process all input tokens **in parallel**

High data reuse across tokens

✓ **Compute-bound**

Generative Inference Has Two Phases

Prefill

Process all input tokens **in parallel**

High data reuse across tokens

✓ **Compute-bound**

then



Decode

Generate **one token** at a time

Read entire model + KV cache

✗ **Memory-bound**

Generative Inference Has Two Phases

Prefill
Process all input tokens **in parallel**
High data reuse across tokens
✓ **Compute-bound**

then

Decode
Generate **one token** at a time
Read entire model + KV cache
✗ **Memory-bound**

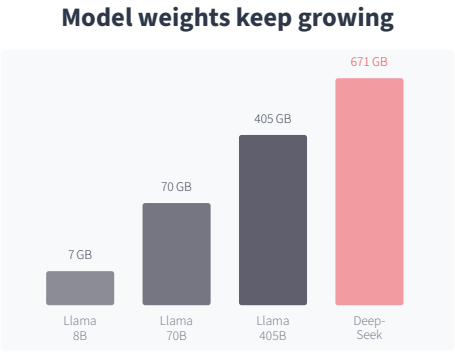
10% | **~90% of wall-clock inference time is spent in decode**

70 GB
model weights (INT8)
read every token

+ KV cache
0.16 MB/token/user
grows linearly with context

0 reuse
full read of weights
and cache, every token

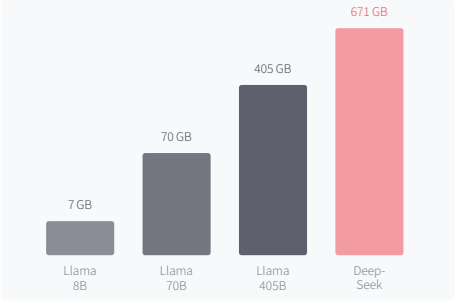
The Growing Data Problem



INT8 weights. Larger models need more cards.

The Growing Data Problem

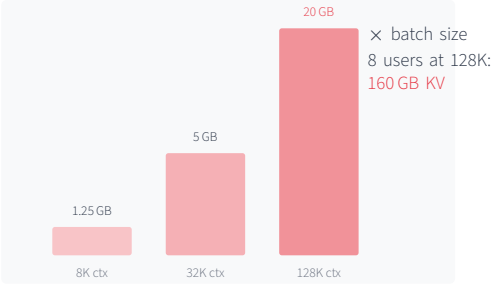
Model weights keep growing



INT8 weights. Larger models need more cards.

KV cache grows with context

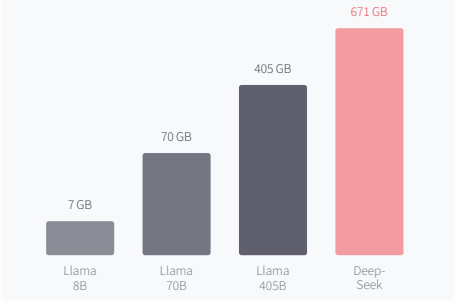
(Llama-3.1 70B, INT8, per user)



Longer context × more users ⇒ KV cache dominates.

The Growing Data Problem

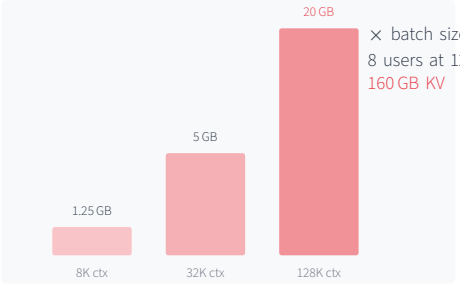
Model weights keep growing



INT8 weights. Larger models need more cards.

KV cache grows with context

(Llama-3.1 70B, INT8, per user)



Longer context × more users ⇒ KV cache dominates.

Weights set the **bandwidth floor**. KV cache sets the **capacity ceiling**. Both are growing.

How Memory-Bound Is Decode?

Arithmetic intensity (AI)
= compute per byte moved

Llama-70B decode:
140 GFLOP ÷ 75 GB = **AI = 1.9**

Balance point (AI*)
= peak compute ÷ memory BW
Below AI*: hardware idles.

	HBM	3D-DRAM
BW	18 TB/s	100 TB/s
AI*	278	50
Gap	146×	26×

More BW → lower AI* → faster

How Memory-Bound Is Decode?

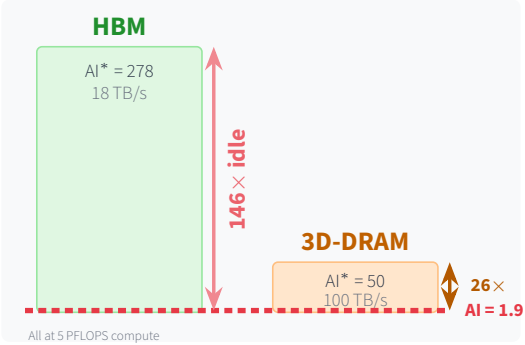
Arithmetic intensity (AI)
= compute per byte moved

Llama-70B decode:
140 GFLOP ÷ 75 GB = **AI = 1.9**

Balance point (AI*)
= peak compute ÷ memory BW
Below AI*: hardware idles.

	HBM	3D-DRAM
BW	18 TB/s	100 TB/s
AI*	278	50
Gap	146×	26×

More BW → lower AI* → faster



Decode operates **146**× below the HBM balance point: the compute units stall on memory bandwidth.

Batching Has Hard Limits

Batching amortizes weight reads

- Batch B users: read weights once, reuse $B\times$
- AI scales: $\mathrm{AI}_B \approx B \cdot \mathrm{AI}_1$

But KV cache scales *with* B

- Memory $=$ 70 GB (weights) $+ B \times \mathrm{KV}_{\text{per user}}$
- 128K context, $B=8$: $70 + 8\times 20 = 230$ GB

Batching Has Hard Limits

Batching amortizes weight reads

- Batch B users: read weights once, reuse $B\times$
- AI scales: $\mathrm{AI}_B \approx B \cdot \mathrm{AI}_1$

But KV cache scales *with* B

- Memory $=$ 70 GB (weights) $+ B \times \mathrm{KV}_{\text{per user}}$
- 128K context, $B=8$: $70 + 8\times 20 = 230$ GB

Batch sizes could be small

Poisson arrival rate	Max B	Avg B
110 req/s	1	1.0
500 req/s	5	1.2
1000 req/s	27	5.1

Batching Has Hard Limits

Batching amortizes weight reads

- Batch B users: read weights once, reuse $B\times$
- AI scales: $\mathrm{AI}_B \approx B \cdot \mathrm{AI}_1$

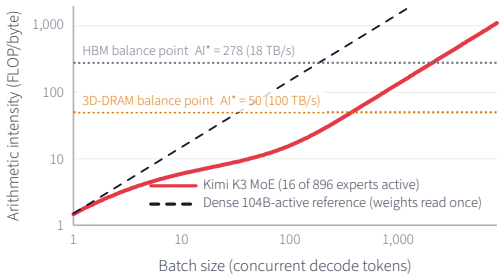
But KV cache scales *with* B

- Memory $=$ **70** GB (weights) $+ B \times \mathrm{KV}_{\text{per user}}$
- 128K context, $B=8$: $70 + 8\times 20 = 230$ GB

Batch sizes could be small

Poisson arrival rate	Max B	Avg B
110 req/s	1	1.0
500 req/s	5	1.2
1000 req/s	27	5.1

Kimi K3 decode: arithmetic intensity vs. batch size



MoE decode: AI rises sub-linearly with batch size (each token activates a subset of experts); AI* $=$ **50** requires $\sim$**400** concurrent tokens.

Batches average $\sim$**5**; MoE decode requires $\sim$**400** to reach AI* = 50.

Batching cannot close the gap.

The decode bottleneck is fundamental. Batching cannot solve it. Longer context makes it worse.

The question becomes:

What kind of memory do we actually need?

Fast enough to feed the compute. Large enough to hold the model.

Today's Memory Technologies: Bandwidth or Capacity, Not Both

SRAM on-die 6T cells: highest bandwidth

Bandwidth	**150 TB/s**
Capacity	**4 GB** per card
I/O energy	∼0.5 pJ/bit latency ∼1 ns

Why capacity cannot scale

- 6T cell is **100–150× larger** than a DRAM 1T1C cell
- ∼1 nW/bit leakage at GB scale
- ∼$1000/GB vs. ∼$10/GB for DRAM

HBM stacked DRAM, 2.5D: highest capacity

Bandwidth	**18 TB/s**
Capacity	**192 GB** per package (8 stacks)
I/O energy	∼2–3 pJ/bit HBM4-class

Why its bandwidth is limited

- mm-scale routing $\Rightarrow$ 2–3 pJ/bit
- 55 μm bump pitch limits I/O density
- A PHY at every stack interface

Today's Memory Technologies: Bandwidth or Capacity, Not Both

SRAM on-die 6T cells: highest bandwidth

Bandwidth	**150 TB/s**
Capacity	**4 GB** per card
I/O energy	∼0.5 pJ/bit latency ∼1 ns

Why capacity cannot scale

- 6T cell is **100–150× larger** than a DRAM 1T1C cell
- ∼1 nW/bit leakage at GB scale
- ∼$1000/GB vs. ∼$10/GB for DRAM

Llama-70B on SRAM

70 GB ≫ 4 GB/card ⇒ **18+ cards** for the weights alone, collectives on every token.

HBM stacked DRAM, 2.5D: highest capacity

Bandwidth	**18 TB/s**
Capacity	**192 GB** per package (8 stacks)
I/O energy	∼2–3 pJ/bit HBM4-class

Why its bandwidth is limited

- mm-scale routing ⇒ 2–3 pJ/bit
- 55 μm bump pitch limits I/O density
- A PHY at every stack interface

Llama-70B on HBM

75 GB/token ÷ 18 TB/s = **4.2 ms** — 8–42% of a 10–50 ms TPOT budget, on reads alone.

Today's Memory Technologies: Bandwidth or Capacity, Not Both

SRAM on-die 6T cells: highest bandwidth

Bandwidth	**150 TB/s**
Capacity	**4 GB** per card
I/O energy	∼0.5 pJ/bit latency ∼1 ns

Why capacity cannot scale

- 6T cell is **100–150× larger** than a DRAM 1T1C cell
- ∼1 nW/bit leakage at GB scale
- ∼$1000/GB vs. ∼$10/GB for DRAM

Llama-70B on SRAM

70 GB ≫ 4 GB/card ⇒ **18+ cards** for the weights alone, collectives on every token.

HBM stacked DRAM, 2.5D: highest capacity

Bandwidth	**18 TB/s**
Capacity	**192 GB** per package (8 stacks)
I/O energy	∼2–3 pJ/bit HBM4-class

Why its bandwidth is limited

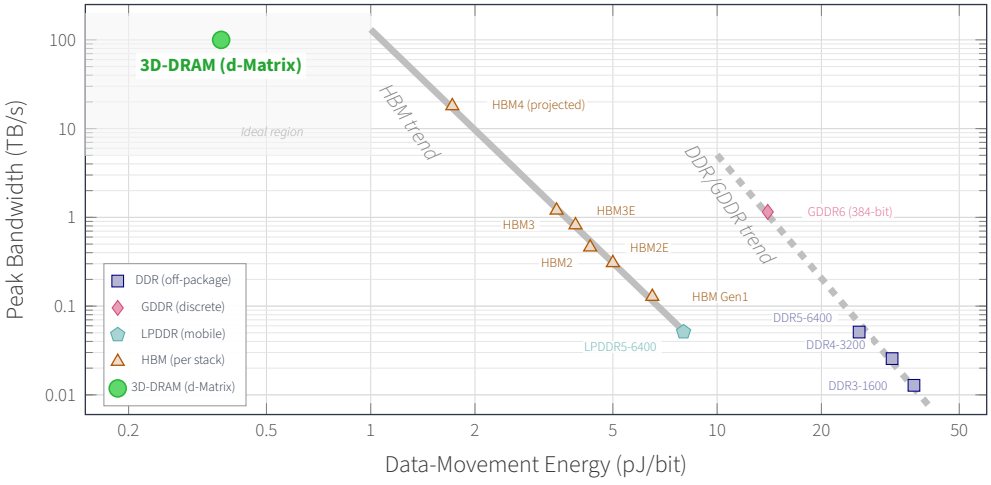
- mm-scale routing ⇒ 2–3 pJ/bit
- 55 μm bump pitch limits I/O density
- A PHY at every stack interface

Llama-70B on HBM

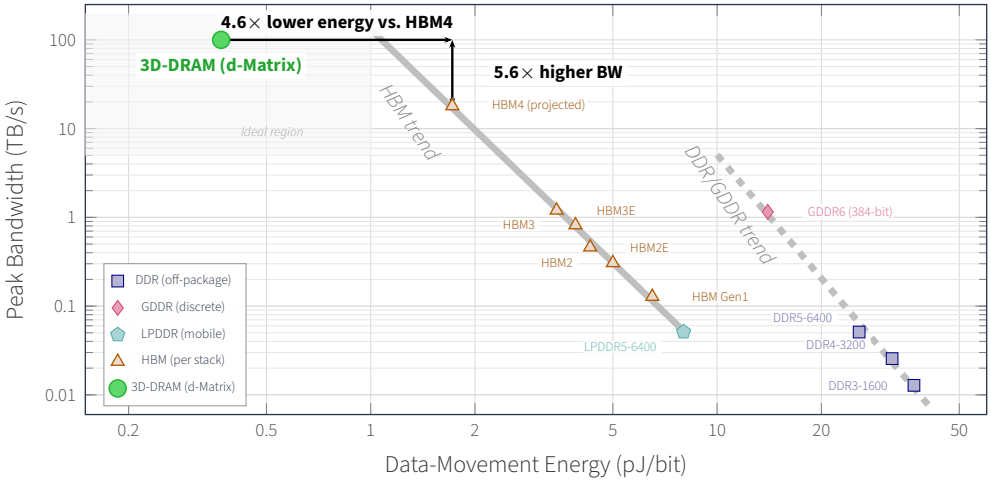
75 GB/token ÷ 18 TB/s = **4.2 ms** — 8–42% of a 10–50 ms TPOT budget, on reads alone.

High bandwidth, small capacity or **large capacity, limited bandwidth**: neither holds a 70B model *and* sustains the compute.

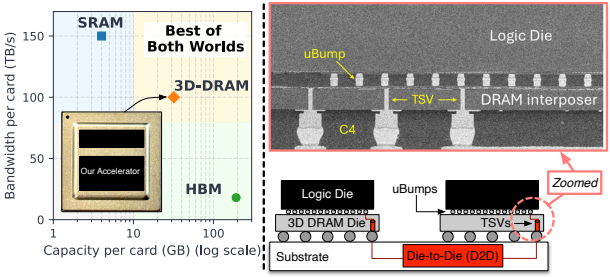
DRAM Landscape: The Power-Normalized Bandwidth (PN) Inversion Law



DRAM Landscape: The Power-Normalized Bandwidth (PN) Inversion Law

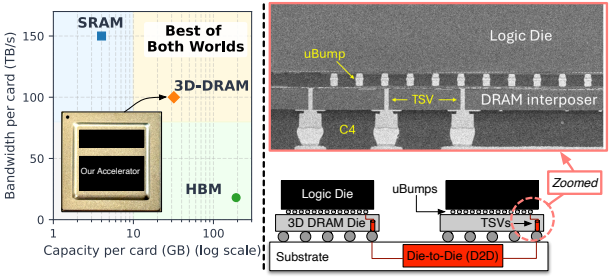


Raptor: The First 3D-DRAM Accelerator for Generative Inference (ISCA 2026)



Logic die bonded face-to-face to 3D-DRAM die.
Vertical interconnects replace mm-scale HBM traces.

Raptor: The First 3D-DRAM Accelerator for Generative Inference (ISCA 2026)



Logic die bonded face-to-face to 3D-DRAM die.
Vertical interconnects replace mm-scale HBM traces.

	HBM	3D-DRAM
Topology	2.5D (beside)	F2F (below)
I/O energy	2–3 pJ/bit	0.37 pJ/bit
BW / card	18 TB/s	100+ TB/s
Capacity	192 GB	32 GB

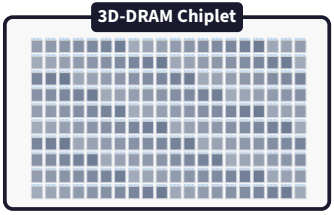
5.6× higher BW

5× lower energy/bit

7× denser I/O

Memory *beneath* the logic rather than beside it:
vertical μbumps replace millimetre-scale interposer traces.

Raptor Silicon: From an 840-Macro Chiplet to a 100 TB/s Card

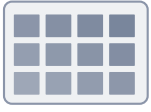


Per chiplet

840	256 DQ	625 MHz	15+ TB/s
macros (72 spare)	per macro	read clock	peak; 4 GB

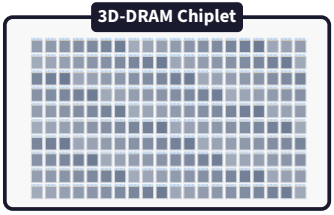
Each macro is a 5.33 MB bank with a dedicated 256-bit port directly into the logic die — no shared bus, no burst, no PHY in between.

Chiplet



840 macros | 4 GB
15+ TB/s

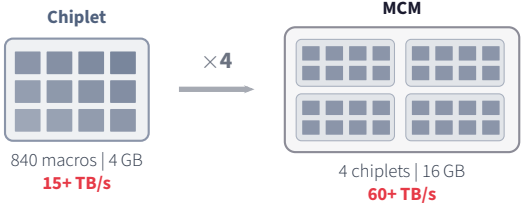
Raptor Silicon: From an 840-Macro Chiplet to a 100 TB/s Card



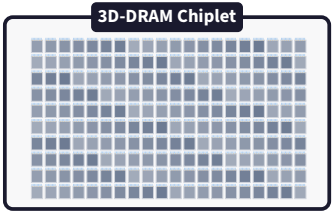
Per chiplet

840	256 DQ	625 MHz	15+ TB/s
macros (72 spare)	per macro	read clock	peak; 4 GB

Each macro is a 5.33 MB bank with a dedicated 256-bit port directly into the logic die — no shared bus, no burst, no PHY in between.



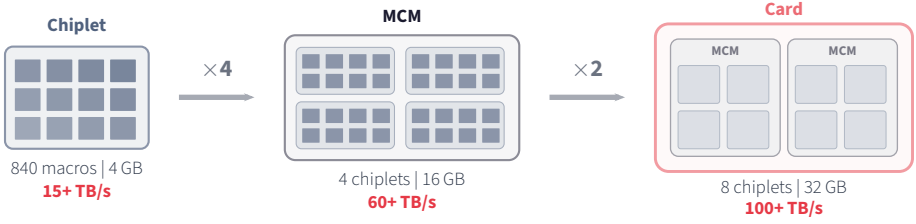
Raptor Silicon: From an 840-Macro Chiplet to a 100 TB/s Card



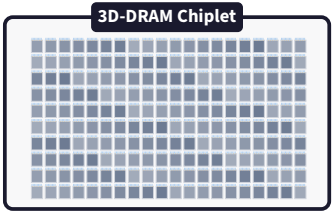
Per chiplet

840	256 DQ	625 MHz	15+ TB/s
macros (72 spare)	per macro	read clock	peak; 4 GB

Each macro is a 5.33 MB bank with a dedicated 256-bit port directly into the logic die — no shared bus, no burst, no PHY in between.



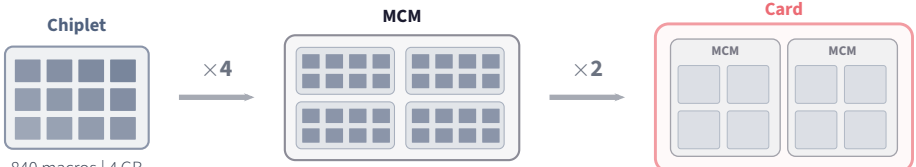
Raptor Silicon: From an 840-Macro Chiplet to a 100 TB/s Card



Per chiplet

840	256 DQ	625 MHz	15+ TB/s
macros (72 spare)	per macro	read clock	peak; 4 GB

Each macro is a 5.33 MB bank with a dedicated 256-bit port directly into the logic die — no shared bus, no burst, no PHY in between.



8 chiplets → 2 MCMs → one card: **32 GB at 100+ TB/s**.
The MCM is the unit of integration at rack scale.

Per mm² of Silicon: Raptor 3D-DRAM vs. HBM4 and Rubin R200

Same silicon area, three memory technologies

Raptor at 83 % utilisation, HBM at 100 %. Rubin R200 = 8 × HBM4 24 Gb. Power per GB/s: lower is better.

Metric	Raptor 3D-DRAM	HBM4 24 Gb	HBM4 32 Gb	Rubin R200
Capacity (MB/mm²)	**11.4**	21.9	26.3	21.9
Bandwidth (GB/s/mm²)	**32.6**	1.67	1.51	1.39
Power per GB/s (mW/GB/s)	**2.96**	40.0	40.0	40.0

Per mm² of Silicon: Raptor 3D-DRAM vs. HBM4 and Rubin R200

Same silicon area, three memory technologies

Raptor at 83 % utilisation, HBM at 100 %. Rubin R200 = 8 × HBM4 24 Gb. Power per GB/s: lower is better.

Metric	Raptor 3D-DRAM	HBM4 24 Gb	HBM4 32 Gb	Rubin R200
Capacity (MB/mm²)	**11.4**	21.9	26.3	21.9
Bandwidth (GB/s/mm²)	**32.6**	1.67	1.51	1.39
Power per GB/s (mW/GB/s)	**2.96**	40.0	40.0	40.0

≈ 20×
the bandwidth per mm² of HBM4

13.5×
better power per GB/s

≈ 0.5×
the capacity per mm²

Per mm² of Silicon: Raptor 3D-DRAM vs. HBM4 and Rubin R200

Same silicon area, three memory technologies

Raptor at 83 % utilisation, HBM at 100 %. Rubin R200 = 8 × HBM4 24 Gb. Power per GB/s: lower is better.

Metric	Raptor 3D-DRAM	HBM4 24 Gb	HBM4 32 Gb	Rubin R200
Capacity (MB/mm²)	**11.4**	21.9	26.3	21.9
Bandwidth (GB/s/mm²)	**32.6**	1.67	1.51	1.39
Power per GB/s (mW/GB/s)	**2.96**	40.0	40.0	40.0
≈ 20× the bandwidth per mm² of HBM4		**13.5×** better power per GB/s		**≈ 0.5×** the capacity per mm²

Per mm²: **20× the bandwidth**, **13.5× the bandwidth per watt**, 0.5× the capacity — the trade-off that favours memory-bound decode.

Bandwidth and capacity in one package. But the DRAM now sits beneath hot logic, under different constraints.

The question becomes:

What breaks when DRAM moves under the logic?

The 3D-DRAM Integration Landscape

Hardware	Architecture	Co-Design
TSV density vs. routing congestion trade-offs	Bank-to-channel mapping: 840 banks $\neq 256 \times 4$	Thermal: T_j=105°C $\Rightarrow$ 8× faster refresh
Power delivery through 3D stack (IR drop, decap)	Weight & KV-cache tiling across banked memory	ECC, DBI metadata, & scrub share bank rows
Signal integrity: crosstalk & SSN on wide parallel bus	Redundancy without breaking channel symmetry	Refresh scheduling vs. compute pipeline bubbles
I/O power: no burst structure $\Rightarrow$ no PHY DBI	LPDDR5X as secondary tier: data placement policy	Known-good-die testing $\rightarrow$ post-package repair flow
μBump yield & alignment at 36 μm pitch	Compiler support for stream-aware access patterns	Multi-high stacking: routing, power, thermals
Clock distribution & skew across die stack	Runtime KV-cache alloc & eviction across banks	Hybrid bonding roadmap: <2 μm pitch, 8-high

The 3D-DRAM Integration Landscape

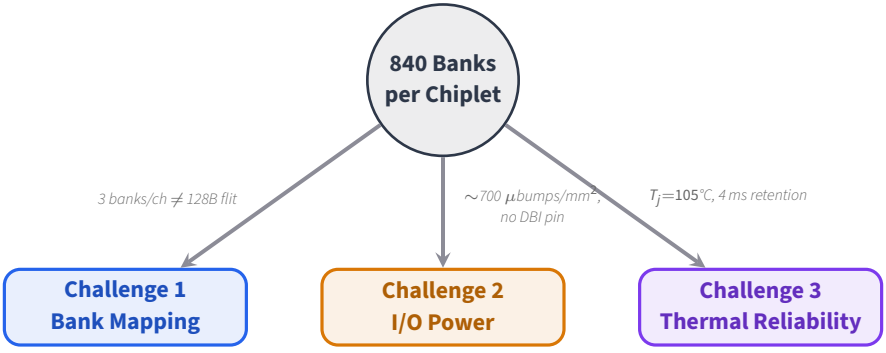
Hardware	Architecture	Co-Design
TSV density vs. routing congestion trade-offs	★ **Bank-to-channel mapping: 840 banks $\neq$ 256$\times$4**	★ **Thermal:** T_j=105°C $\Rightarrow$ **8$\times$ faster refresh**
Power delivery through 3D stack (IR drop, decap)	Weight & KV-cache tiling across banked memory	ECC, DBI metadata, & scrub share bank rows
Signal integrity: crosstalk & SSN on wide parallel bus	★ **Redundancy without breaking channel symmetry**	Refresh scheduling vs. compute pipeline bubbles
★ **I/O power: no burst structure $\Rightarrow$ no PHY DBI**	LPDDR5X as secondary tier: data placement policy	Known-good-die testing $\rightarrow$ post-package repair flow
μBump yield & alignment at 36 μm pitch	Compiler support for stream-aware access patterns	Multi-high stacking: routing, power, thermals
Clock distribution & skew across die stack	Runtime KV-cache alloc & eviction across banks	Hybrid bonding roadmap: $<$2 μm pitch, 8-high

The 3D-DRAM Integration Landscape

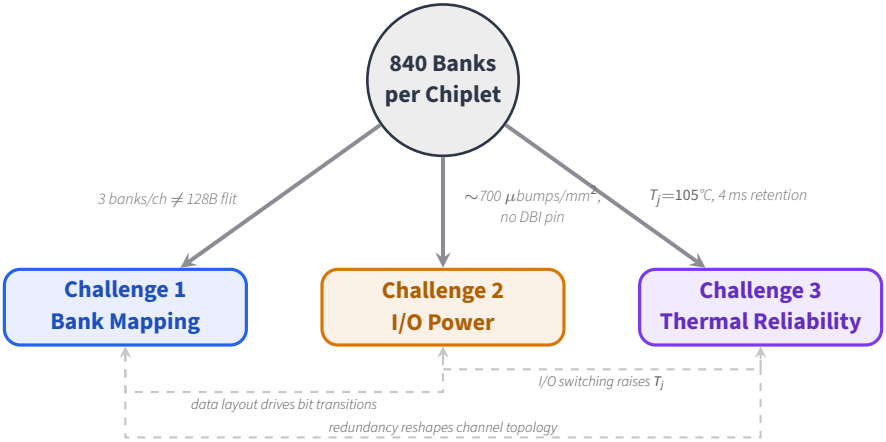
Hardware	Architecture	Co-Design
TSV density vs. routing congestion trade-offs	★ **Bank-to-channel mapping: 840 banks $\neq$ 256$\times$4**	★ **Thermal: T_j=105°C $\Rightarrow$ 8$\times$ faster refresh**
Power delivery through 3D stack (IR drop, decap)	Weight & KV-cache tiling across banked memory	ECC, DBI metadata, & scrub share bank rows
Signal integrity: crosstalk & SSN on wide parallel bus	★ **Redundancy without breaking channel symmetry**	Refresh scheduling vs. compute pipeline bubbles
★ **I/O power: no burst structure $\Rightarrow$ no PHY DBI**	LPDDR5X as secondary tier: data placement policy	Known-good-die testing $\rightarrow$ post-package repair flow
μBump yield & alignment at 36 μm pitch	Compiler support for stream-aware access patterns	Multi-high stacking: routing, power, thermals
Clock distribution & skew across die stack	Runtime KV-cache alloc & eviction across banks	Hybrid bonding roadmap: $<$2 μm pitch, 8-high

Integrating DRAM beneath the logic opens a broad problem space —
★ marks the problems this talk addresses: **three entangled challenges**.

Three Entangled Integration Challenges



Three Entangled Integration Challenges



A solution to any one challenge **constrains the design space of the other two.**

Challenge 1: The Bank-to-Channel Mapping Problem



The setup

- Each TE needs a **128 B flit** per access
- 16 ch × 16 TE-Groups = **256 ch/chiplet**
- Each bank delivers **32 B** per column access

Challenge 1: The Bank-to-Channel Mapping Problem



The setup

- Each TE needs a **128 B flit** per access
- 16 ch × 16 TE-Groups = **256 ch/chiplet**
- Each bank delivers **32 B** per column access

The arithmetic

- 128 B ÷ 32 B = **4** banks/ch *needed*
- DRAM die: **840 banks** (768 after 72 spares)
- 768 ÷ 256 = **3** banks/ch *available*

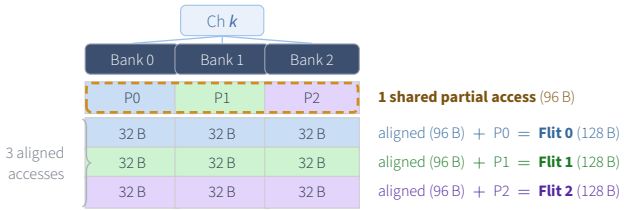
Need 4 banks/channel ⇒ have only 3 ⇒ **the 128 B flit does not divide evenly**:
two 96 B accesses fetch 192 B per 128 B flit ⇒ **33% overfetch ≈ 33 TB/s wasted**

Challenge 1 ✓ Solution: Stream Blocking

Challenge. A 128 B flit = **4**×**32 B**, but each channel has only **3 banks** — a naïve fetch reads 192 B per 128 B flit (⇒ **33% overfetch**).

Challenge 1 ✓ Solution: Stream Blocking

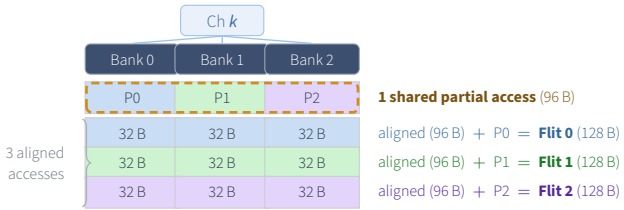
Challenge. A 128 B flit = **4×32 B**, but each channel has only **3 banks** — a naïve fetch reads 192 B per 128 B flit (⇒ **33% overfetch**).



Puts the 4th 32 B (the *partial*) into one access *shared* by 3 flits — $4 \times 96 = 384$ B in $= 3 \times 128 = 384$ B out ⇒ **0% overfetch**.

Challenge 1 ✓ Solution: Stream Blocking

Challenge. A 128 B flit = **4×32 B**, but each channel has only **3 banks** — a naïve fetch reads 192 B per 128 B flit (⇒ **33% overfetch**).



Puts the 4th 32 B (the *partial*) into one access *shared* by 3 flits — $4 \times 96 = 384$ B in $= 3 \times 128 = 384$ B out ⇒ **0% overfetch**.

Every column access is fully utilised — the **33 TB/s** lost to overfetch is reclaimed, with **no shifting network**.

Challenge 2: The I/O Power Wall

The power cost of 100 TB/s:

Measured I/O Power

$P_{I/O} = 100\ \text{TB/s} \times 0.45\ \text{pJ/bit} = \mathbf{360}\ \text{W}$

Why conventional DBI cannot help:

- HBM uses multi-cycle bursts (BL 4–16)
- PHY sees full burst $\Rightarrow$ decides on inversion
- DBI pin per byte lane signals choice

Our 3D-DRAM has **none of these**: single-cycle transfer, no burst, no sideband pin.

Challenge 2: The I/O Power Wall

The power cost of 100 TB/s:

Measured I/O Power

$P_{I/O} = 100\ \text{TB/s} \times 0.45\ \text{pJ/bit} = \mathbf{360}\ \text{W}$

Why conventional DBI cannot help:

- HBM uses multi-cycle bursts (BL 4–16)
- PHY sees full burst ⇒ decides on inversion
- DBI pin per byte lane signals choice

Our 3D-DRAM has **none of these**: single-cycle transfer, no burst, no sideband pin.

Interface comparison:

	DDR / HBM
Transfer	Multi-cycle burst
Lookahead	Full burst visible
DBI	Dedicated pin/lane

	Our 3D-DRAM
Transfer	Single-cycle 256-bit
Lookahead	**None**
DBI	**No pin. No sideband.**

DBI would save ∼20%

∼70 W per card — interface must be rethought.

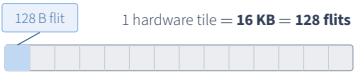
How to achieve DBI **architecturally?**

Challenge 2 ✓ Solution: Stream Flipping (Pinless DBI)

Challenge. I/O dominates: $100\,\text{TB/s} \times 0.45\,\text{pJ/bit} = \mathbf{360\,W}$, and conventional DBI requires a **sideband pin** that the single-cycle 3D-DRAM link lacks.

Challenge 2 ✓ Solution: Stream Flipping (Pinless DBI)

Challenge. I/O dominates: 100 TB/s × 0.45 pJ/bit = **360 W**, and conventional DBI requires a **sideband pin** that the single-cycle 3D-DRAM link lacks.



⇒ **long** per-channel flit streams; each flit compared to the *previous* one.

(a) Without DBI

flit $i-1$: 0 0 · · · 0
flit i: 1 1 · · · 1
flit $i+1$: 0 0 · · · 0

→ every bit toggles twice

(b) Stream flipping: invert flit i

0 0 · · · 0	**0**
0 0 · · · 0 (was 1 · · · 1)	**1**
0 0 · · · 0	**0**

meta

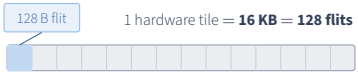
→ ≈0 toggles + 1 metadata bit / flit

Architectural DBI, no pin:

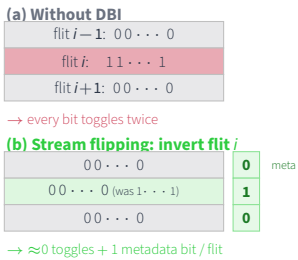
- **Write:** compare each flit to the previous → invert.
- **Read:** fetch the 1-bit tag; invert if set
- Tag co-located with ECC ⇒ **0.8%** overhead, **no PHY change**.

Challenge 2 ✓ Solution: Stream Flipping (Pinless DBI)

Challenge. I/O dominates: 100 TB/s × 0.45 pJ/bit = **360 W**, and conventional DBI requires a **sideband pin** that the single-cycle 3D-DRAM link lacks.



⇒ **long** per-channel flit streams; each flit compared to the *previous* one.

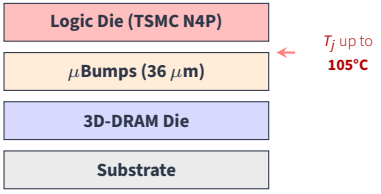


Architectural DBI, no pin:

- **Write:** compare each flit to the previous → invert.
- **Read:** fetch the 1-bit tag; invert if set
- Tag co-located with ECC ⇒ **0.8%** overhead, **no PHY change**.

Stream flipping: **0.45** → **0.37 pJ/bit** (−18%), 360 W → **296 W** per card — with **no pin and no PHY change**.

Challenge 3: DRAM Reliability at 105°C



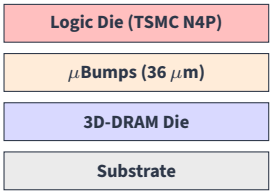
Retention consequence

Standard: 32 ms at 85°C

At 105°C: drops to **4 ms**

$\Rightarrow$ **8**$\times$ more frequent refresh

Challenge 3: DRAM Reliability at 105°C



Retention consequence

Standard: 32 ms at 85°C

At 105°C: drops to **4 ms**

⇒ **8**× more frequent refresh

Three compounding problems:

1. Yield — 840 banks/die. Even a 1% fault rate risks losing entire channels. Discarding bonded dies is uneconomical.

2. Symmetry — Disabling a faulty bank narrows its channel. Tensor engines expect *uniform* channel widths—one weak channel throttles the entire slice.

3. Error accumulation — Higher T_j drives more soft errors. ECC, scrub, and refresh must all coexist without stalling throughput.

How to maintain full-bandwidth, symmetric channels when banks fail and refresh is **8**× nominal?

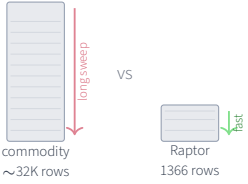
Challenge 3 ✓ Solution: Thermal-Aware Refresh & ECC

Challenge. At **105°C**, retention falls 32 ms → 4 ms (**8**× more refresh), and bits still need correcting — all *without losing significant bandwidth*.

Challenge 3 ✓ Solution: Thermal-Aware Refresh & ECC

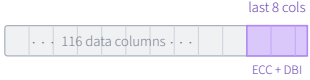
Challenge. At **105°C**, retention falls 32 ms → 4 ms (**8×** more refresh), and bits still need correcting — all *without losing significant bandwidth*.

Deep banking makes refresh inexpensive



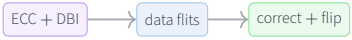
16–32× fewer rows ⇒ 4 ms refresh (8× more frequent) costs only **1.37%** → **~103 TB/s**.

Interleaved ECC, co-located with DBI



Last **8 of 124** columns: paired **[132,128]** Reed–Solomon + DBI bits, interleaved with the subarray.

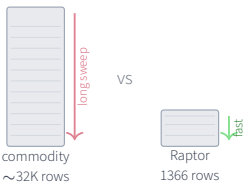
Read order:



Challenge 3 ✓ Solution: Thermal-Aware Refresh & ECC

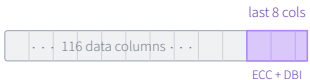
Challenge. At **105°C**, retention falls 32 ms → 4 ms (**8**× more refresh), and bits still need correcting — all *without losing significant bandwidth*.

Deep banking makes refresh inexpensive



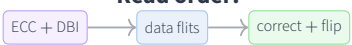
16–32× fewer rows ⇒ 4 ms refresh (8× more frequent) costs only **1.37%** → **~103 TB/s**.

Interleaved ECC, co-located with DBI



Last **8 of 124** columns: paired **[132,128]** Reed–Solomon + DBI bits, interleaved with the subarray.

Read order:



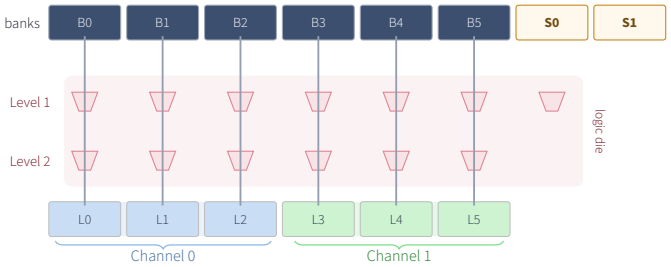
Deep banking turns an 8× refresh penalty into only a **<1.4% bandwidth loss**

Challenge 3 ✓ Solution: Bank Chaining

Challenge. The **72 spare banks** must cover faults *anywhere* on the die — without making channels asymmetric or adding long edge wires.

Challenge 3 ✓ Solution: Bank Chaining

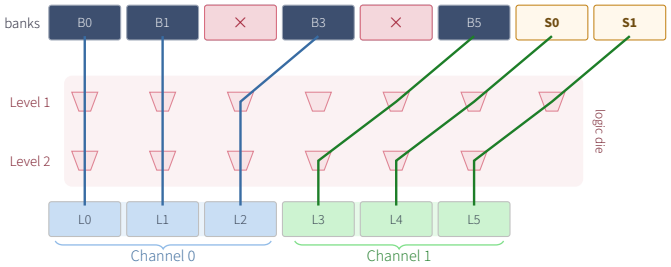
Challenge. The **72 spare banks** must cover faults *anywhere* on the die — without making channels asymmetric or adding long edge wires.



No faults — both mux levels pass each bank straight to its slot.

Challenge 3 ✓ Solution: Bank Chaining

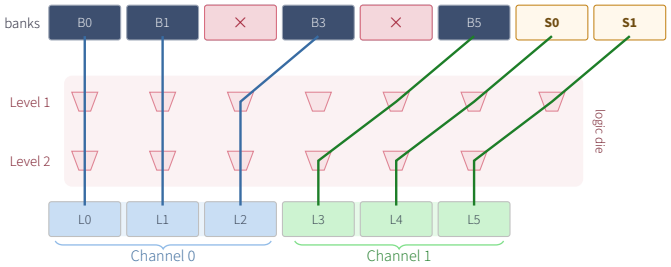
Challenge. The **72 spare banks** must cover faults *anywhere* on the die — without making channels asymmetric or adding long edge wires.



*Level 1 skips the first fault (B2); Level 2 skips the second (B4) — **M=2 physical mux levels tolerate 2 faults anywhere**, spares backfill, channels stay symmetric.*

Challenge 3 ✓ Solution: Bank Chaining

Challenge. The **72 spare banks** must cover faults *anywhere* on the die — without making channels asymmetric or adding long edge wires.



*Level 1 skips the first fault (B2); Level 2 skips the second (B4) — **M=2 physical mux levels tolerate 2 faults anywhere**, spares backfill, channels stay symmetric.*

The chiplet's **72 in-line redundant macros** absorb faults anywhere — symmetric channels, negligible routing cost.

One card: 32 GB at 100 TB/s. *A frontier model: hundreds of GB of weights,* *terabytes of KV cache.*

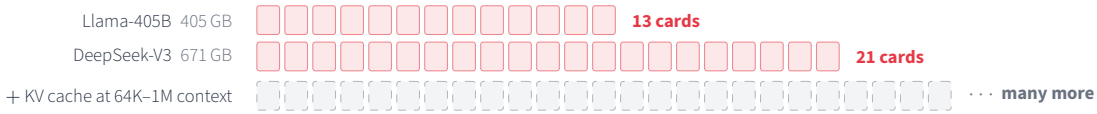
The question becomes:

What does a balanced 3D-DRAM rack look like?

Capacity to hold the model. Interconnect to sustain the bandwidth. Cooling to preserve reliability.

Scaling Out: The Parallelism Tax

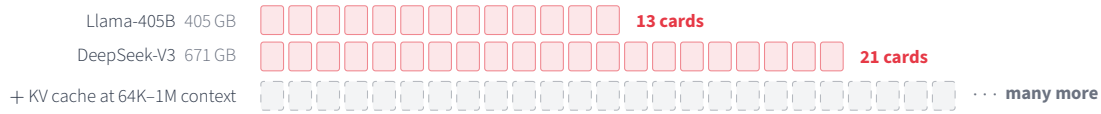
A frontier model does not fit on one card INT8 weights only; one card = 32 GB



Scaling Out: The Parallelism Tax

A frontier model does not fit on one card

INT8 weights only; one card = 32 GB



The model is partitioned across MCMs — three strategies, each with its own collective communication

Tensor parallel



Split every weight matrix: **2 all-reduces per layer** on every token's critical path.

Pipeline parallel



Split layers into stages: one activation hop per stage; **bubbles** if unbalanced.

Expert parallel

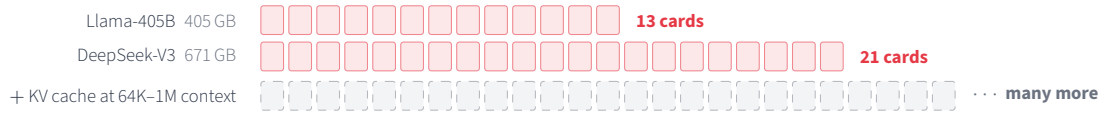


Route tokens to experts (MoE): **2 all-to-alls per MoE layer**, dispatch and combine.

Scaling Out: The Parallelism Tax

A frontier model does not fit on one card

INT8 weights only; one card = 32 GB



The model is partitioned across MCMs — three strategies, each with its own collective communication

Tensor parallel



Split every weight matrix: **2 all-reduces per layer** on every token's critical path.

Pipeline parallel



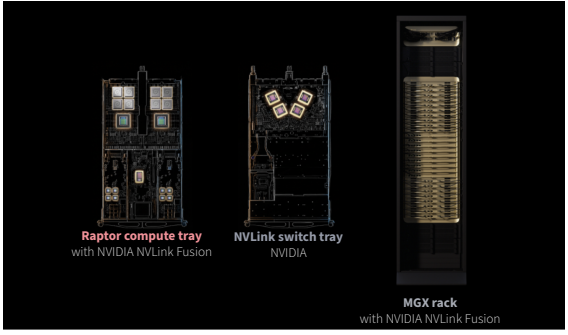
Split layers into stages: one activation hop per stage; **bubbles** if unbalanced.

Expert parallel



Route tokens to experts (MoE): **2 all-to-alls per MoE layer**, dispatch and combine.

Anatomy of a Heterogeneous AI Rack



d-Matrix Raptor trays and MGX rack with NVIDIA NVLink Fusion.

Compute tray (1U)
the unit of compute

8 Raptor MCMs (= 4 cards): **128 GB**, **400 TB/s**. Liquid-cooled cold plates → 1U form factor.

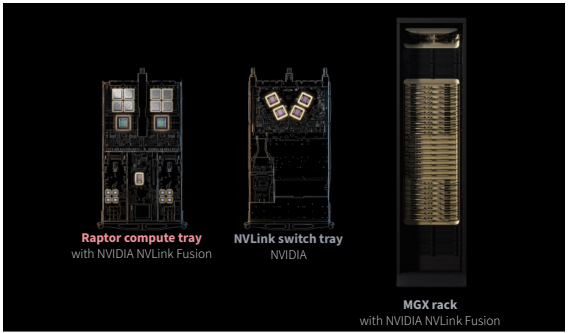
Scale-up switch tray
the rack's internal network

High-radix switches: any MCM reaches any other MCM in **one hop**; every collective (all-reduce, all-to-all) traverses them.

Rack
the unit of deployment

18 compute trays and power shelves; ∼**200 kW**, → liquid-cooled.

Anatomy of a Heterogeneous AI Rack



d-Matrix Raptor trays and MGX rack with NVIDIA NVLink Fusion.

Compute tray (1U)
the unit of compute

8 Raptor MCMs (= 4 cards): **128 GB**, **400 TB/s**. Liquid-cooled cold plates → 1U form factor.

Scale-up switch tray
the rack's internal network

High-radix switches: any MCM reaches any other MCM in **one hop**; every collective (all-reduce, all-to-all) traverses them.

Rack
the unit of deployment

18 compute trays and power shelves; ∼**200 kW**, → liquid-cooled.

Scale-up: intra-rack, one hop, µs latency, TB/s. **Scale-out**: inter-rack, over NICs.
Scale-up domain size and latency bound the achievable performance.

The Raptor Rack by the Numbers

144	**7.2 PB/s**	**2.3 TB**	**200 kW**
MCMs: 18 trays × 8	memory bandwidth	3D-DRAM capacity	power envelope

Design rules: 1U trays, enabled by liquid cooling · **one hop** between any two MCMs (switch radix is the hard constraint) · NICs for scale-out.

144 × 50 TB/s sustained = 7.2 PB/s; 144 × 16 GB = 2.3 TB.

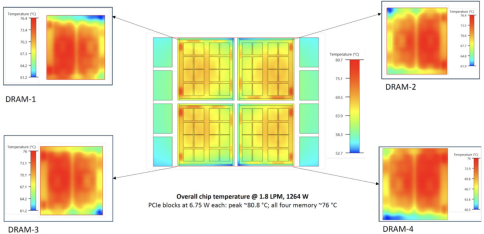
The Raptor Rack by the Numbers

144	**7.2 PB/s**	**2.3 TB**	**200 kW**
MCMs: 18 trays × 8	memory bandwidth	3D-DRAM capacity	power envelope

Design rules: 1U trays, enabled by liquid cooling · **one hop** between any two MCMs (switch radix is the hard constraint) · NICs for scale-out.

144 × 50 TB/s sustained = 7.2 PB/s; 144 × 16 GB = 2.3 TB.

Cold plates designed for the 3D-DRAM hotspots



- One MCM at **1.26 kW**, 1.8 L/min coolant
- All four DRAM dies ≈ **76 °C**; hottest block ≈ 81 °C
- ∼30 °C of margin below the **105 °C** design point of Challenge 3

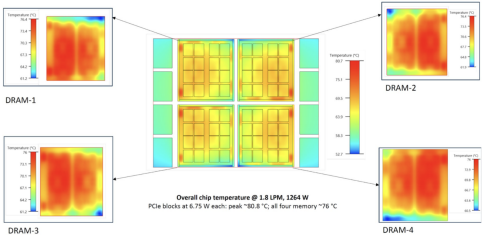
The Raptor Rack by the Numbers

144	**7.2 PB/s**	**2.3 TB**	**200 kW**
MCMs: 18 trays × 8	memory bandwidth	3D-DRAM capacity	power envelope

Design rules: 1U trays, enabled by liquid cooling · **one hop** between any two MCMs (switch radix is the hard constraint) · NICs for scale-out.

144 × 50 TB/s sustained = 7.2 PB/s; 144 × 16 GB = 2.3 TB.

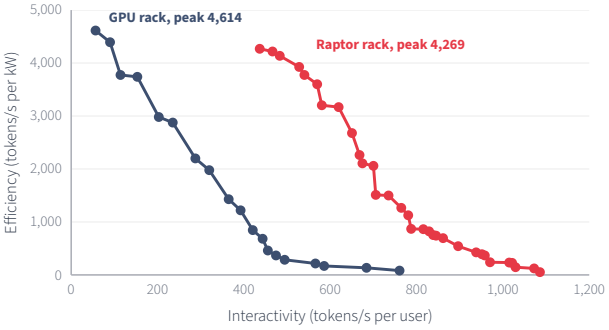
Cold plates designed for the 3D-DRAM hotspots



- One MCM at **1.26 kW**, 1.8 L/min coolant
- All four DRAM dies ≈ **76 °C**; hottest block ≈ 81 °C
- ∼30 °C of margin below the **105 °C** design point of Challenge 3

Dense, single-hop, liquid-cooled:

Kimi K3 at 64K Context: The Raptor Rack Shifts the Pareto Frontier

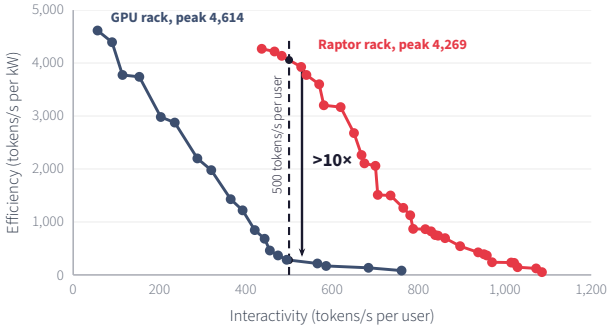


Kimi K3 (MoE), 64K context, simulated; interactivity = **1000**/TPOT (ms). Frontiers are Pareto-optimal over batch size and deployment.

Reading the chart

- *x*-axis: **interactivity**, tokens/s per user
- *y*-axis: **efficiency**, tokens/s per kW
- Larger batch ⇒ higher efficiency, lower interactivity

Kimi K3 at 64K Context: The Raptor Rack Shifts the Pareto Frontier



Kimi K3 (MoE), 64K context, simulated; interactivity = 1000/TPOT (ms). Frontiers are Pareto-optimal over batch size and deployment.

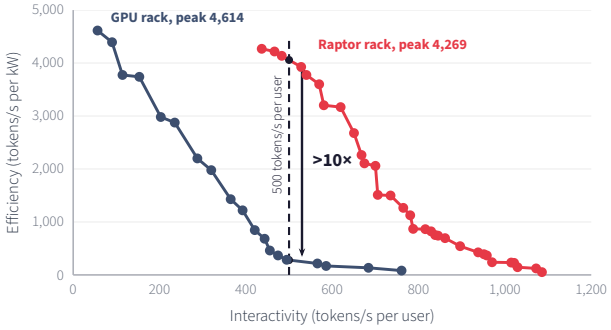
Reading the chart

- *x*-axis: **interactivity**, tokens/s per user
- *y*-axis: **efficiency**, tokens/s per kW
- Larger batch ⇒ higher efficiency, lower interactivity

What it shows

- **Peak efficiency is comparable**: 4.6k vs. 4.3k tokens/s per kW
- **The frontier extends** to ∼**1,100** vs. ∼750 tokens/s per user (∼1.5× the interactivity)
- **At 500 tokens/s per user** (dashed line): >**10**× the efficiency

Kimi K3 at 64K Context: The Raptor Rack Shifts the Pareto Frontier



Kimi K3 (MoE), 64K context, simulated; interactivity = 1000/TPOT (ms). Frontiers are Pareto-optimal over batch size and deployment.

Reading the chart

- *x*-axis: **interactivity**, tokens/s per user
- *y*-axis: **efficiency**, tokens/s per kW
- Larger batch ⇒ higher efficiency, lower interactivity

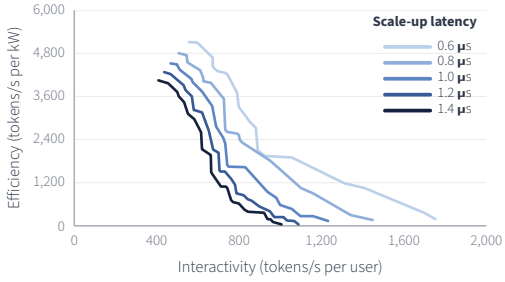
What it shows

- **Peak efficiency is comparable**: 4.6k vs. 4.3k tokens/s per kW
- **The frontier extends** to ∼**1,100** vs. ∼750 tokens/s per user (∼1.5× the interactivity)
- **At 500 tokens/s per user** (dashed line): >**10**× the efficiency

At long context the KV cache dominates the memory traffic — 7.2 PB/s of rack bandwidth converts it into **interactivity** rather than stall time.

Rack Connectivity: Latency and Bandwidth Are First-Order

Kimi K3 on **4**× 144-MCM Raptor racks, 64K context, simulated — same axes as before, sweeping the scale-up network



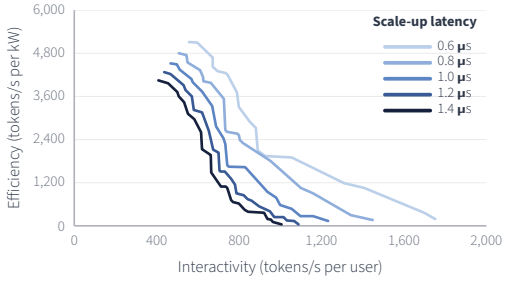
Network latency: 0.6 → 1.4 µs

Interactivity ceiling: ∼1,750 → ∼1,000 tokens/s per user.

Every collective incurs it ⇒ single-hop, high-radix switching.

Rack Connectivity: Latency and Bandwidth Are First-Order

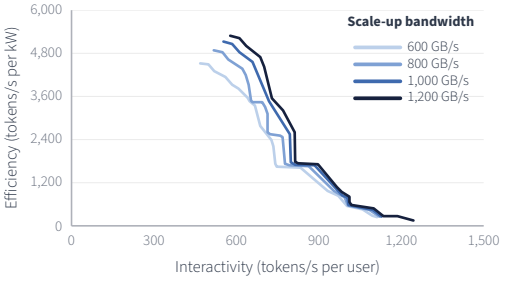
Kimi K3 on **4**× 144-MCM Raptor racks, 64K context, simulated — same axes as before, sweeping the scale-up network



Network latency: 0.6 → 1.4 µs

Interactivity ceiling: ∼1,750 → ∼1,000 tokens/s per user.

Every collective incurs it ⇒ single-hop, high-radix switching.



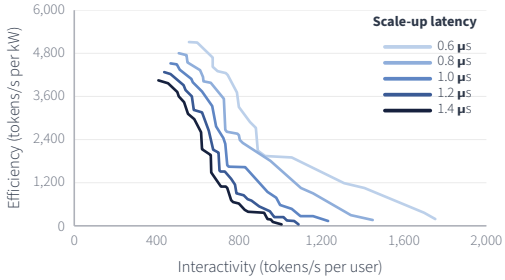
Network bandwidth: 600 → 1200 GB/s

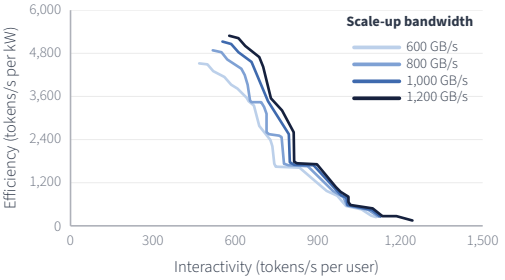
Peak efficiency: ∼4.5k → ∼5.3k tokens/s per kW.

Larger batches and models ⇒ greater bandwidth sensitivity.

Rack Connectivity: Latency and Bandwidth Are First-Order

Kimi K3 on **4**× 144-MCM Raptor racks, 64K context, simulated — same axes as before, sweeping the scale-up network





Network latency: 0.6 → 1.4 µs

Interactivity ceiling: ∼1,750 → ∼1,000 tokens/s per user.

Every collective incurs it ⇒ single-hop, high-radix switching.

Network bandwidth: 600 → 1200 GB/s

Peak efficiency: ∼4.5k → ∼5.3k tokens/s per kW.

Larger batches and models ⇒ greater bandwidth sensitivity.

3D-DRAM bandwidth is realised only with a matching scale-up network:
single-hop, low-latency, high-bandwidth.

Conclusions and Outlook

The silicon

- Decode is **memory-bound**; batching cannot close the gap.
- **3D-DRAM face-to-face**: 100 TB/s, 32 GB, 0.37 pJ/bit per card.
- **Stream blocking, stream flipping, bank chaining**: co-designed for 105 °C.

The rack

- **144 MCMs**: 7.2 PB/s, 2.3 TB, 200 kW, liquid-cooled.
- **Kimi K3**: 1.5× the interactivity, >10× the efficiency at 500 tokens/s per user.
- **The scale-up network is first-order**: single hop, low latency, high bandwidth.

Outlook

- **Stacking**: hybrid bonding at <2 µm, 8-high ⇒ 56× bandwidth per watt.
- **Interconnect**: a 576-MCM single-hop scale-up domain.
- **Heterogeneity**: attention on GPUs, experts on 3D-DRAM.

Conclusions and Outlook

The silicon

- Decode is **memory-bound**; batching cannot close the gap.
- **3D-DRAM face-to-face**: 100 TB/s, 32 GB, 0.37 pJ/bit per card.
- **Stream blocking, stream flipping, bank chaining**: co-designed for 105 °C.

The rack

- **144 MCMs**: 7.2 PB/s, 2.3 TB, 200 kW, liquid-cooled.
- **Kimi K3**: 1.5× the interactivity, >10× the efficiency at 500 tokens/s per user.
- **The scale-up network is first-order**: single hop, low latency, high bandwidth.

Outlook

- **Stacking**: hybrid bonding at <2 µm, 8-high ⇒ 56× bandwidth per watt.
- **Interconnect**: a 576-MCM single-hop scale-up domain.
- **Heterogeneity**: attention on GPUs, experts on 3D-DRAM.

The memory wall is a technology problem. **3D-DRAM is the solution** — and the rack is where it is realised.

Scaling the Memory Wall

From Early 3D-DRAM Silicon to Heterogeneous AI Racks

Thank You