Long Live the Short King: Why 4-hi HBM Wins

newsletter.semianalysis.com/p/long-live-the-short-king-why-4-hi

Myron Xie, Bryan Shan, Harrison Barclay, Minjae Kang, Dylan Patel September 13, 2026

High Bandwidth Memory has been a key technology enabling the AI revolution. Despite HBM's high costs relative to other forms of memory, chip designers have packaged more and more HBM into AI accelerators. Customers push to design in newer generation HBM whilst also increasing capacity per XPU by adding more cubes, and with denser and higher stacks. This has led to HBM consuming an increasing share of total DRAM wafer capacity, resulting in the extreme DRAM shortage we see ourselves in today.

We are now on the cusp of this trend breaking, due to a host of reasons that are both technical and supply chain driven. Next generation accelerators are moving to 8-hi stacks as standard compared to 12-hi being the standard today. Nvidia has decided to do this with Rubin Ultra which is a downgrade to 192GB from 288GB per GPU compared to standard Rubin and B300, which SemiAnalysis was the first to cover this change. The supply chain is now preparing 8-hi to become the new standard, and it was not even a year ago that the industry was anticipating going to 16-hi and beyond.

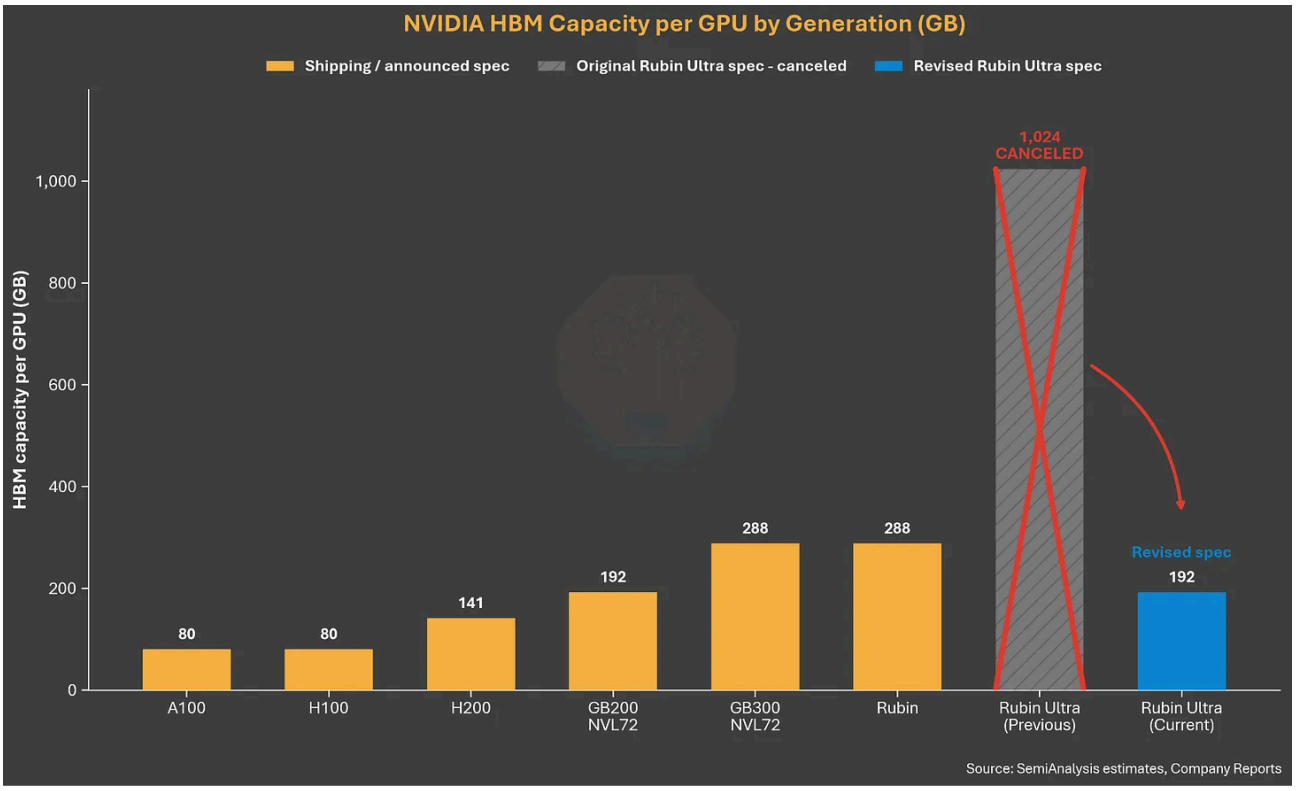
However, we see this trend going further as we covered in our Memory Model. For inference workloads where memory bandwidth matters the most, 4-hi HBM provides the best $/bandwidth and therefore lowest cost per token. Capacity is important up until a certain threshold, beyond that point more HBM capacity offers diminishing benefits while those incremental bits carry the same BOM penalty. This is what hardware teams at the major labs want to implement in their ASIC programs from next generation HBM4 onwards. On the same way that hardware designers are focussed on maximizing tokens/Watt in a world of DC power constraints, 4-hi is how to maximize tokens per HBM wafer, which is another scarce resource.

This report will go through the reasoning behind what's changed in the capacity equation, demonstrate that often HBM capacity is stranded, and show how this can be a win-win for both labs' TCO and memory supplier profitability.

The HBM Content Trend is Broken

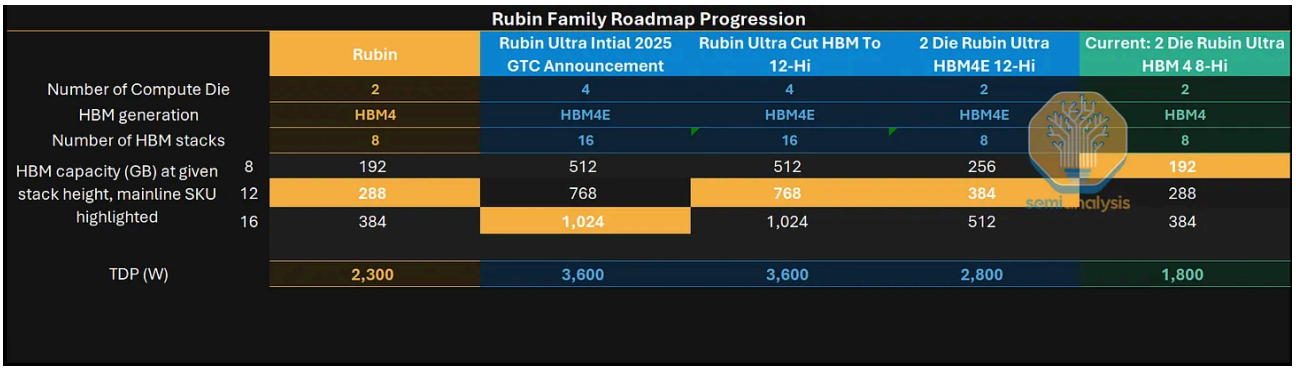
HBM has been the major form of working memory for AI workloads as it provides a good balance of capacity and bandwidth which can be utilized for general purpose chips for both training and inference. SRAM, while very fast, suffers from low capacity, making it premium and limited on certain tasks. Conventional DDR DRAM provides much more capacity per dollar but the bandwidth is too poor.

With AI being such a memory intensive workload, it is no wonder that per accelerator HBM content has continued to increase generation upon generation: more cubes per accelerator, denser cubes, and taller stacks.



Source: SemiAnalysis, Nvidia

The increasing amount of HBM being shipped per chip was also a big contributor to costs increasing. However, over the last few months this trend has now broken. The next generation Rubin Ultra will only feature 192GB of HBM per accelerator compared to 288GB with conventional Rubin and even Blackwell Ultra! This is a huge regression from expectations from not even a year ago that Rubin Ultra would feature 1TB of HBM per GPU! In fairness, that was when Rubin Ultra was expected to have 4 compute dies vs only 2 now. Even normalizing for this, this is a decrease from 256GB per GPU die to only 96GB.



		Rubin	Rubin Ultra Intial 2025 GTC Announcement	Rubin Ultra Cut HBM To 12-Hi	2 Die Rubin Ultra HBM4E 12-Hi	Current: 2 Die Rubin Ultra HBM 4 8-Hi
Number of Compute Die		2	4	4	2	2
HBM generation		HBM4	HBM4E	HBM4E	HBM4E	HBM4
Number of HBM stacks		8	16	16	8	8
HBM capacity (GB) at given stack height, mainline SKU highlighted	8	192	512	512	256	192
	12	288	768	768	384	288
	16	384	1,024	1,024	512	384
TDP (W)		2,300	3,600	3,600	2,800	1,800

Source: SemiAnalysis, Nvidia

This decision to reduce HBM density is in large part motivated by supply. HBM wafers are scarce. There are simply not enough wafers to supply the number of 12-hi cubes needed

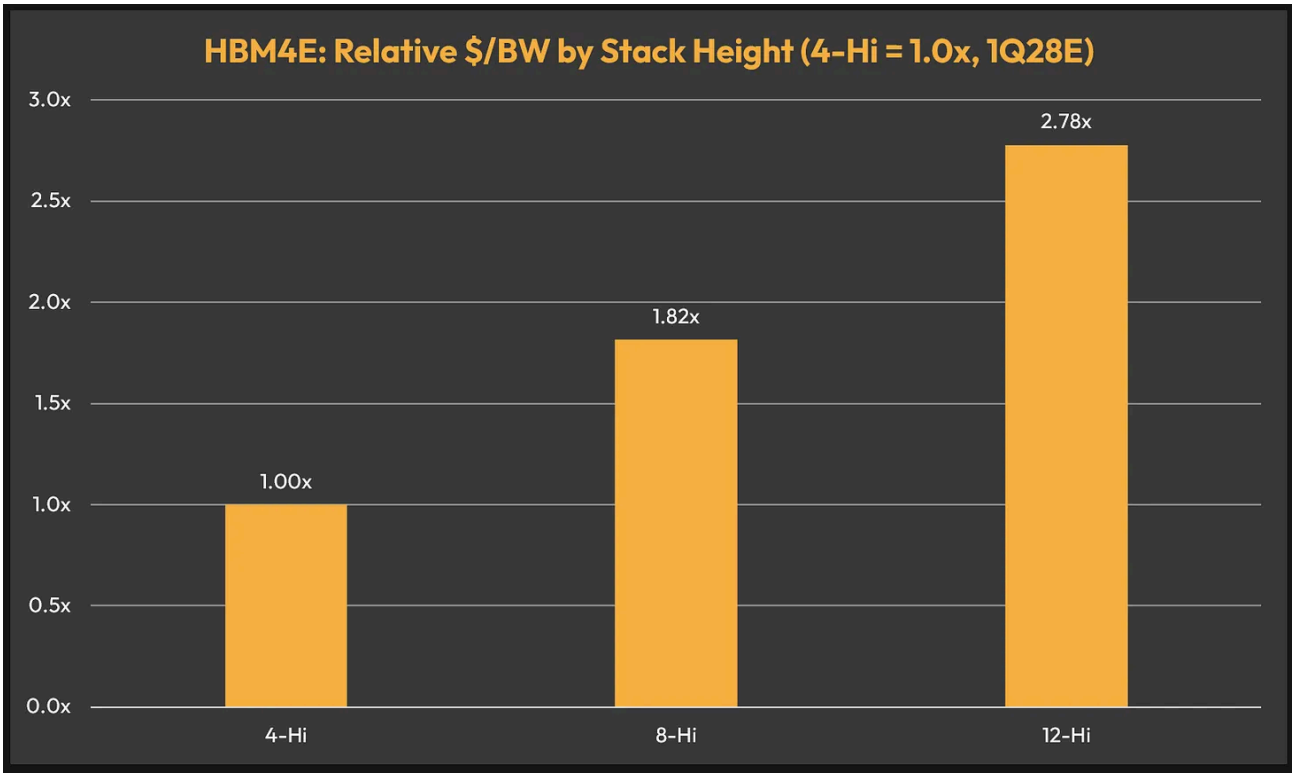
to support the N3 logic and CoWoS Nvidia has secured next year. Moving to 8-hi means much more cubes from the limited number of wafers and helps close the gap.

The big increase in HBM costs next year doesn't help either, even though Nvidia is getting a better deal than other major customers. Even in a time where GPU demand is well above supply, Nvidia has to give up some margin as its component costs increase, with memory being the largest.

The trend of ever increasing HBM density is now breaking and it is not only because of supply, but also architectural reasons and changing workload requirements. While supply and cost are very important factors, Nvidia wouldn't make this decision if this additional capacity is necessary for customers and clearly delivers better performance/TCO. The focus of this article is to demonstrate that in many cases additional HBM capacity per GPU is not worth the cost and that HBM 4-hi in the future is often the optimal configuration.

Shorter Stacks, Same Bandwidth

An important dynamic we want to remind readers of is that the reduction in per GPU capacity by lowering stack height doesn't come at the expense of bandwidth at all. Bandwidth available per cube of HBM is the same regardless of stack height. Of the 2048 data I/Os per cube of HBM4 or HBM4e, these data paths are split evenly among the total number core DRAM dies in the stack. The maximum per die is 512 I/Os, which means that 4-hi is the lowest stack height that can get access to all of the 2048 I/Os and maximize bandwidth. However, cost to the supplier and end customer is driven primarily by the amount of DRAM content per cube. HBM pricing is primarily driven by GB per stack. While the value customers derive from HBM is often the bandwidth. Therefore lower stack heights mean a much lower $/bandwidth proposition to end customers, meaning opting for lower stack heights is almost a free lunch.



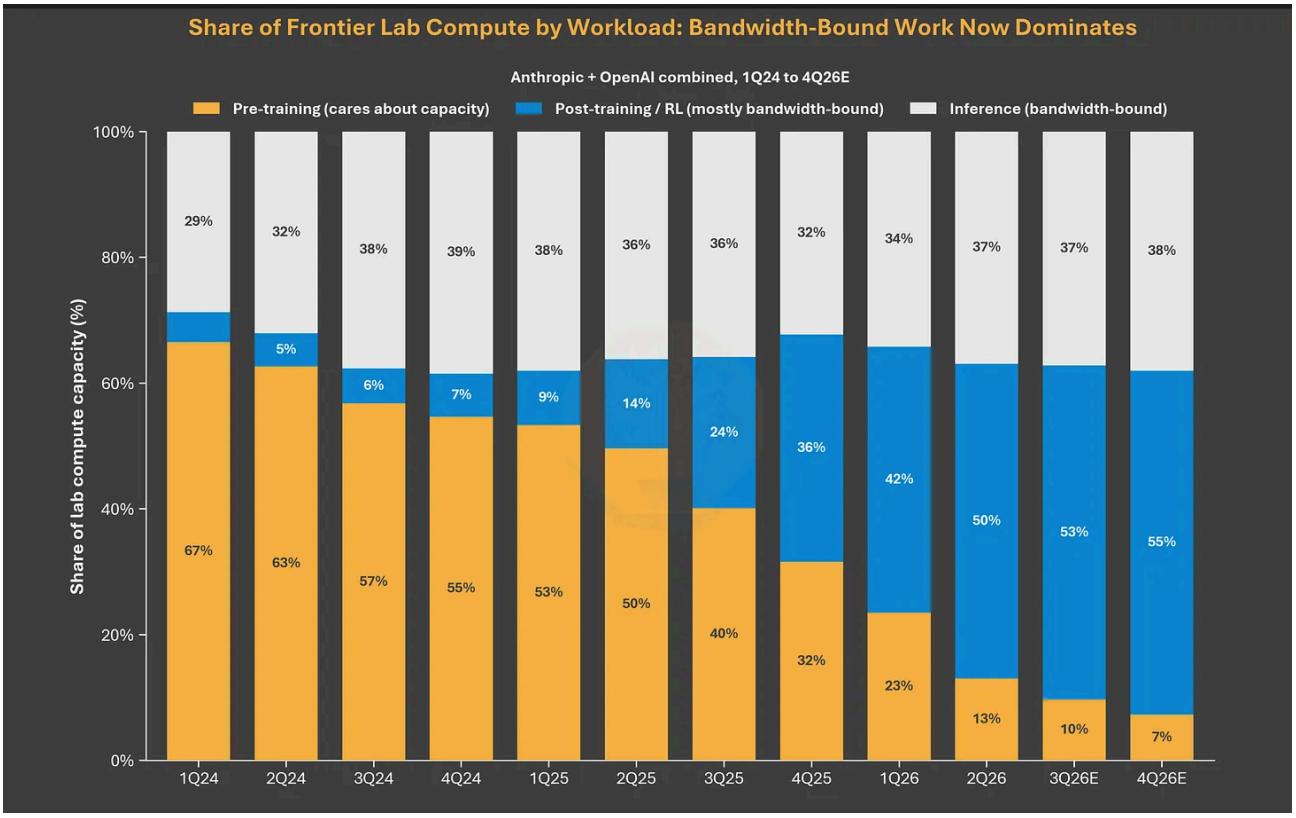
Source: SemiAnalysis

Bandwidth vs Capacity? Bandwidth is King

While different AI workloads require both capacity and bandwidth, there are tilts towards whether bandwidth or capacity matter more depending on workload. We can divide AI compute into 3 major buckets: pre-training compute, post-training/reinforcement learning compute, inference compute. Reinforcement learning is very similar to inference therefore requires similar architectures. Among these 3 buckets, inference decode is primarily bandwidth constrained: each token generated requires a read of all the active model parameters as well as the user's KVCache that is stored in memory. Reinforcement Learning (which is inference-like) is still tilted towards being bandwidth constrained but requires more capacity than inference due to the focus on large batches. Meanwhile pre-training is much more sensitive to capacity as HBM is used to store weights, gradients and activations during a training run, rather than just weights and user KV for an inference forward pass.

A big change has been the share of compute allocated to pre-training has shrunk significantly as end token demand has increased and post-training techniques became a major vector for model scaling. The result is that the majority of allocated compute has now become bandwidth sensitive rather than capacity sensitive.

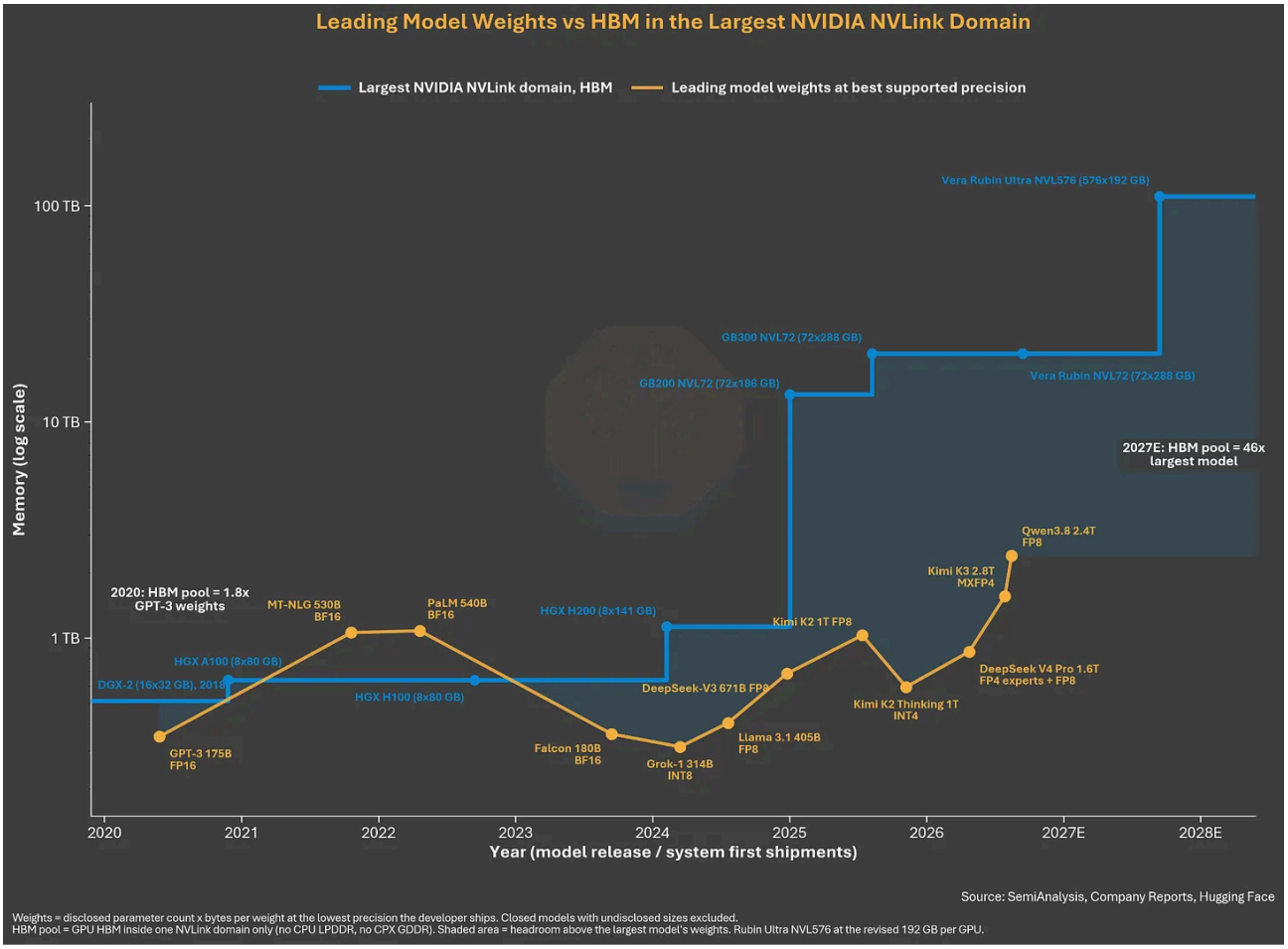




Source: SemiAnalysis Tokenomics Model

This is not to say that capacity is irrelevant for inference, capacity is important up to a point. One of the big weaknesses of the Cerebras and Groq SRAM machines is their lack of capacity despite abundant SRAM bandwidth allowing very fast token serving. In short, there is a threshold level of capacity required: an inference system needs to have enough high bandwidth memory capacity to hold model weights as well as users' KVCache. Beyond that threshold there are diminishing returns to having more capacity as we will show.

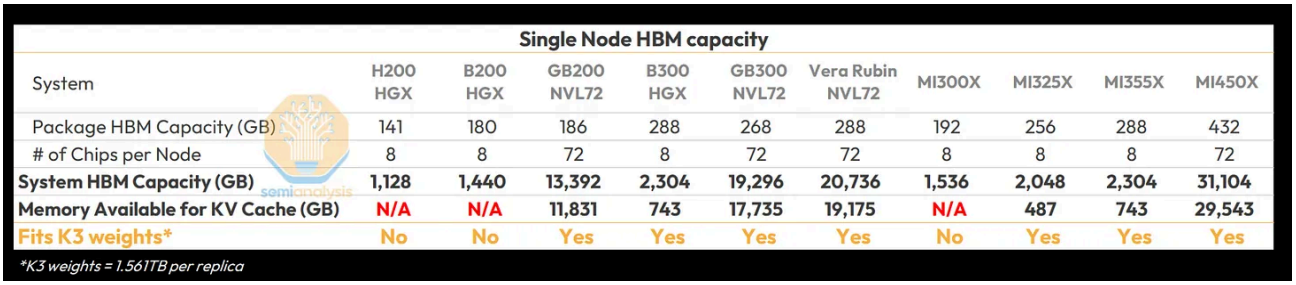
The other important point is that this is not memory per chip, but rather aggregate memory per system. Frontier model sizes have long stopped being able to fit on the memory of a single chip, and need to be sharded across the memory of multiple chips. This is why networking is such an integral part of AI system design. For the MoE architecture that all frontier models of today adopt, the optimal system is often a single scale up world with fast and low-latency communications (such as NVLink within a whole rack) that can support wide expert parallelism, where only a few experts are loaded onto each GPU. This set up allows each expert to process larger token batches and improves compute and memory bandwidth utilization, at the expense of far more all-to-all communication costs between GPUs



Source: SemiAnalysis, Nvidia, Hugging Face

While both model weights and user KVCache capacity requirements have grown, the huge increase in scale up world sizes have allowed aggregate memory per node to more than keep up. In the Hopper-era, one H100 HGX system provided 640GB of aggregate HBM. The weights from Llama 3.1 405B (the best open source model at that time) would consume 63% of that capacity, leaving some (but not a lot of) headroom for KVCache. This was why the additional 61GB per GPU that the H200 provided was so important. With 1,128GB of memory per server, this provided a lot of incremental headroom to fit more KVCache and serve more concurrent users.

With GB300 NVL72 rack scale solutions, Nvidia delivered 9x the scale up world size in addition to doubling HBM per GPU by using 12-hi HBM and increasing the number of HBM cubes per GPU. This resulted in an 18x increase in aggregate HBM capacity compared to the H200 HGX. This has significantly outpaced the almost 7x growth in parameters of the largest open source model available today: Kimi K3 2.8T. The capacity burden is also reduced significantly by quantizing down to 4 bit quarter precision from 16/8 bit. One replica of Kimi K3 at 2.8T stored in MXFP4 only takes up 1,561GB, **less than 8%** of the almost 21TB of HBM available on GB300 NVL72. The increase in scale up world size is also not slowing down. The big upgrade with Rubin Ultra is moving to NVL576, another 8x increase from NVL72, which more than offsets the one third reduction in HBM capacity per GPU.



Single Node HBM capacity										
System	H200 HGX	B200 HGX	GB200 NVL72	B300 HGX	GB300 NVL72	Vera Rubin NVL72	MI300X	MI325X	MI355X	MI450X
Package HBM Capacity (GB)	141	180	186	288	268	288	192	256	288	432
# of Chips per Node	8	8	72	8	72	72	8	8	8	72
System HBM Capacity (GB)	**1,128**	**1,440**	**13,392**	**2,304**	**19,296**	**20,736**	**1,536**	**2,048**	**2,304**	**31,104**
Memory Available for KV Cache (GB)	**N/A**	**N/A**	**11,831**	**743**	**17,735**	**19,175**	**N/A**	**487**	**743**	**29,543**
Fits K3 weights*	**No**	**No**	**Yes**	**Yes**	**Yes**	**Yes**	**No**	**Yes**	**Yes**	**Yes**

**K3 weights = 1.561TB per replica*

Source: SemiAnalysis, Moonshot, Nvidia, AMD

In the rack-scale server era and beyond, memory capacity available is well above what is necessary for model weights, providing a very comfortable amount of headroom for user KVCache. Systems can now comfortably support inference requests with very large context lengths and at good enough concurrency to support very high inference margins. This eases a lot of the HBM capacity pressures and allows HBM density per chip to relax. This is a different paradigm from the previous era of smaller serving deployments where model weights consumed a substantial portion of total memory. Back then increasing HBM density per GPU was a critical way to get valuable capacity headroom.

While the lower stacks, cheaper bandwidth argument has been a valid one, we are only now seeing this trend of taller stacks reverse as the minimum HBM capacity per chip threshold has only recently moved much lower. We think this trend goes even further. Silicon and hardware teams at frontier labs see 4-hi HBM as the optimal SKU for many inference workloads as it delivers the best performance/TCO and now we will dive deeper into why.

The Optimal Mix of Bandwidth and Capacity

There is an optimal ratio of bandwidth and capacity, and often this capacity ends up being over-provisioned or stranded where the bandwidth per token is not enough to read all the available capacity in HBM.

Let's start with a simple example. Each HBM4E stack provides 3,328 GB/s of nominal peak bandwidth (assuming 2048 pins at 13 Gbps each). For HBM4E each core die is 32-Gb which results in 16 GB, 32 GB, and 48 GB for 4-hi, 8-hi, and 12-hi stacks, respectively. An illustrative single sequence of 100 tokens/s, 3,328/100=33.28GB of memory bandwidth is available per token. We then divide this bandwidth available per token by cube capacity to get to the number of reads of the total stack capacity each token is able to perform. At 1 stack read per token this means capacity and bandwidth are matched perfectly: one token has the available bandwidth to read all the working data stored in HBM and no more and no less.

A number above 1 means bandwidth is "overprovisioned" relative to capacity, as a token only needs one read of the working set (i.e. one forward pass). However, the bandwidth is rarely "excess" as the additional bandwidth budget can be used up by increasing tokens

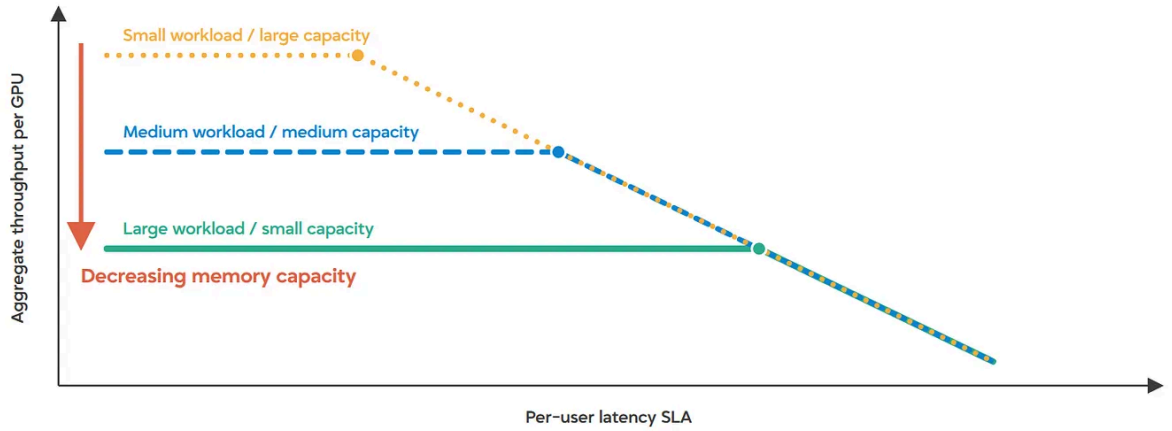
per second. A number below 1 means that a token doesn't have available bandwidth to read all the available HBM capacity. To serve tokens at that given throughput the working set can be no larger than available bandwidth per token, so it must be lower than total memory capacity. This results in excess memory capacity that is overprovisioned.

Bandwidth Budget per Token for HBM4E 3.328TB/s 4-Hi/8-Hi/12-Hi				
		Stack reads per token		
Interactivity	Bandwidth budget per token	4-Hi (16 GB)	8-Hi (32 GB)	12-Hi (48 GB)
100 tokens/s	33.3 GB/token	2.08	1.04	0.69
200 tokens/s	16.6 GB/token	1.04	0.52	0.35
400 tokens/s	8.3 GB/token	0.52	0.26	0.17

Source: SemiAnalysis

As we can see in the illustrative table above, an increase in tokens/s lowers bandwidth available per token, pushes the stack reads/token ratio lower, therefore increasing the likelihood that capacity is overprovisioned. This creates the asymmetry where it is preferable to have a lot of bandwidth compared to a lot of capacity. Bandwidth can normally be used up in its entirety, while spare capacity can not always be used up whilst also carrying a cost that is only getting higher.

Of course, this is a simple analysis that ignores batching requests to serve multiple concurrent users. Batching plays a key role in inference economics as batching can maximize total token throughput at given levels of interactivity. The big saving from batching comes from amortizing one read of weights across multiple users' requests. Higher concurrency also consumes memory capacity as each individual user's KVcache needs to be loaded into memory. Increasing batch size is often how inference workloads become memory capacity bounded rather than bandwidth bounded. When HBM capacity is easily enough to hold model weights, the benefit of additional HBM capacity is that it can deliver higher token throughput, and why the additional bits can be worth the BOM.



Source: SemiAnalysis

When factoring in batching, the result is this generalized curve above. All else equal, having taller stacks of HBM delivers higher throughput up until a certain per-user latency SLA, whereby the extra capacity becomes redundant. These are represented by the points where different curves overlap with each other.

The “per-user latency SLA” is a minimum interactivity assumption. This is a minimum rather than actual interactivity. In cases when concurrency is capped by capacity that means that there is additional bandwidth available to support higher tokens/s/user. So even though 4-hi can deliver lower concurrency, the faster token/s/user wholly or partially offsets the impact of lower batch size to increase total token throughput.

Concurrency is based on the lower of maximum concurrency when capacity-bound: the number of sets of individual users’ KVCache that can fit into memory; and when bandwidth-bound: maximum number of users that can be served at our minimum interactivity, which takes into account the bandwidth available per token calculation above. Multiplying users by tokens/s/user gets to total tokens generated per second which is throughput.

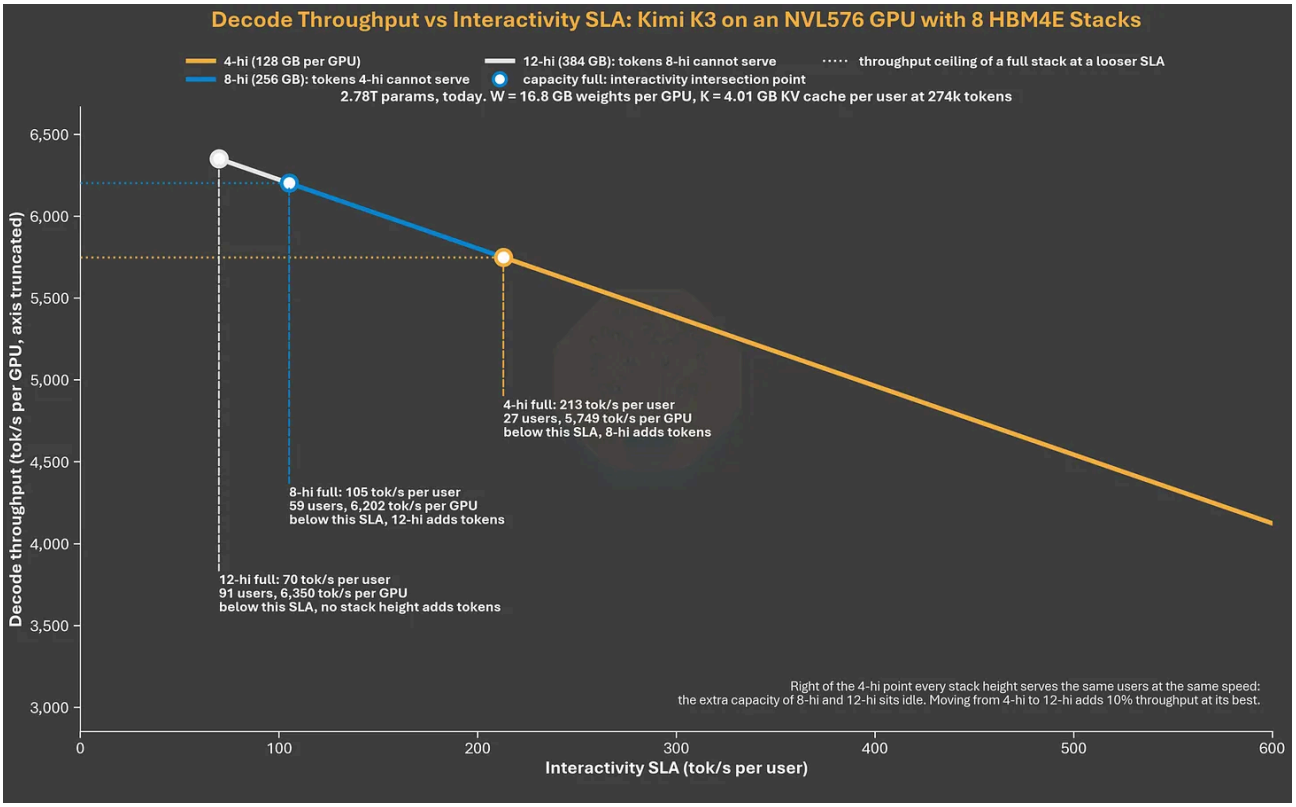
These curves can tell us the value of that additional capacity provides in terms of inference performance by showing just how much greater token throughput can be achieved with more capacity (which comes through more batching), while also showing at which points on minimum interactivity the additional capacity doesn’t add value.

This is heavily dependent on model size or said another way the % of HBM capacity that needs to be dedicated to model weights. In general, the larger the model size, the curves will shift to the right so that taller stacks deliver more throughput at higher interactivity

points. In addition, taller stacks will also deliver a greater increase in peak throughput. This goes back to batching economics. Every step reads the weights once regardless of batch size, so the larger the weight footprint, the more throughput each additional concurrent user unlocks.

Let's put actual numbers to this to see what this looks like for the largest open source model today: Kimi K3. This is a roofline analysis, in reality, real achieved memory bandwidth is significantly lower than peak bandwidth. Real results will see the minimum interactivity thresholds shifted to the left, so the general conclusions will be the same. Also this exercise is focussed on just the decode pool in a disaggregated PD set up. For real delivered results see our InferenceX and AgentX benchmarks.

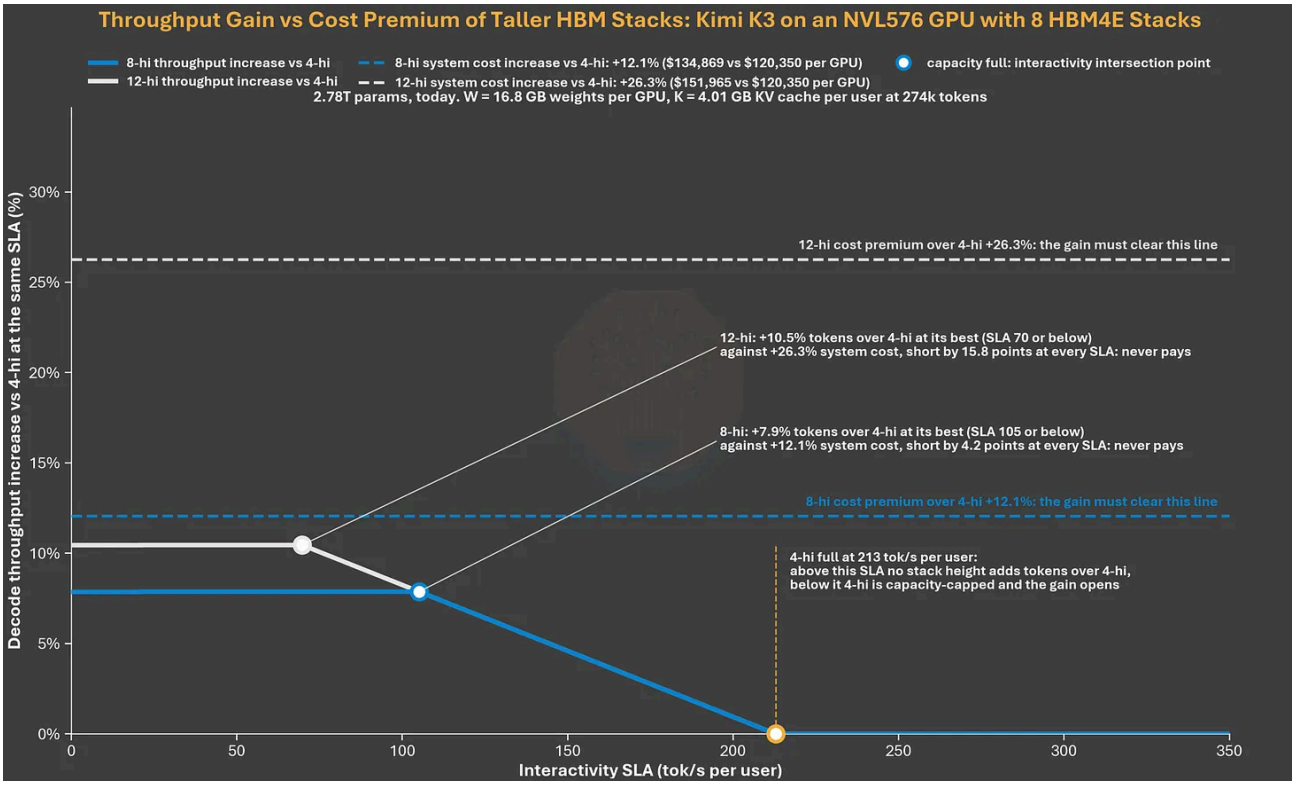
We use a Rubin Ultra NVL576 system as this will be a more representative scale up world size in the HBM4E era. We assume the HBM is HBM4E with 13Gbps speeds. 4-hi, 8-hi, and 12-hi means 128GB, 256GB, and 384GB per GPU respectively. For Kimi K3 on this NVL576 system, each GPU holds 16.8GB of weights which is 2.1GB per HBM stack. We can see that at a minimum SLA of 105 tok/s/user, 12-hi HBM doesn't deliver more throughput than 8-hi HBM. Then when the interactivity is doubled further to 213 tok/s/user, there is no benefit to having taller than 4-hi stacks. In terms of maximum throughput benefits, 8-hi provides an 8% increase in maximum total tokens/s vs 4-hi while 12-hi provides a 10% improvement. However, again stepping down to the maximum throughput that 4-hi can deliver allows for more than double the interactivity. We also reiterate that in reality, as not all of the raw memory bandwidth can be utilized, actual results will shift these curves to the left, meaning that the higher capacity breakeven interactivity points are lower and could push into the speeds that users find unacceptably low.



Source: SemiAnalysis, Moonshot

We can see that while more capacity can deliver marginally more token throughput for these models (putting aside the benefit of faster tokens). The question then becomes, is that additional throughput worth the additional BOM cost associated with more memory. HBM is a significant part of accelerator BOM cost, and with HBM so tight, the cost per GB will rise dramatically next year.

When taking into account costs, we model the BOM cost increase the end owner pays relative to a baseline HBM4E 4-hi based Rubin Ultra for the Rubin Ultra NVL576 system. The all-in system cost increase (which we proxy as TCO as all-in system cost makes up the vast majority of TCO) of 8-Hi is 12.1% relative to a base 4-hi system and for 12-hi it becomes a 26.3% premium. We can then compare the increase in token throughput relative to the cost increase at different points on minimum interactivity to show whether more memory can deliver lower cost per token at any of those points.



Source: SemiAnalysis

Overall we see that the increase in throughput provided by 12-hi or 8-hi HBM comes in below the increased system cost associated with more memory, meaning at these memory prices (along with Nvidia's margin stack), 12-Hi and 8-Hi HBM result in higher cost per token than 4-hi HBM. This comes before we even take into account more HBM conserving techniques, namely KVCache offloading to lower tiers of memory. This next section will show an actual production run we performed to show how lower memory impacts actual throughput.

A Real Experiment: KVCache Offloading Relieves Capacity Pressure

While reducing HBM capacity cuts into the amount of KVCache that is stored as long as model weights are able to be fit, second tier DRAM can be used to store KVCache that is less "hot" and requires instant retrieval. This is becoming more common in agentic workloads that have very high context lengths that place pressure on HBM capacity but also have long pauses between turns, making it worthwhile to expel KVCache into DRAM or even storage rather than recomputing it altogether. In our recent AgentX benchmarking, many workloads' pareto optimal results were obtained with KVCache offloading above certain concurrencies.

To test the impact of reduced HBM in a real workload, we ran InferenceX benchmarks on Kimi K3 with reduced HBM. The system was with 16 GB300 GPUs, with a disaggregated setup of 8 GPUs for the prefill pool and 8 GPUs for the decode pool. The restricted memory workload was run with 85% HBM utilization compared to 92% HBM utilization

normally. Yes, this is only an 8% reduction in HBM, which is very far from a 50% reduction when going from 8-hi to 4-hi, however this is with Kimi K3 being served on only 8 GPUs so weights take up a significant portion of memory. This ends up being a 20GB reduction per GPU of KVCache storage. Specifically, for aggregate serving it brings down the GPU KV budget from 53GB to 34GB which is a 36% reduction. For disaggregated pairs it is 44GB to 24GB, which is a 44% reduction. Overall, this is a sizeable reduction in the HBM budget for KVCache.

In terms of results, we see that up to most points on the curve the HBM restricted run delivered very similar throughput (denoted by the dotted red curve). However once concurrency hit above 70 is when throughput dropped dramatically by almost 30% compared to the higher HBM run. This is the point when peak GPU KV occupancy hit 100% for the restricted memory version. At that point preemption, repeated loading, and queueing affect that throughput. Note that this also hits at a point of the curve that is high concurrency but very low interactivity. We know that DRAM offloading was used much more as the lower HBM-restricted profile had over 10x the amount of reads to DRAM KVCache compared to the higher HBM profile at the same concurrency.



Source: SemiAnalysis

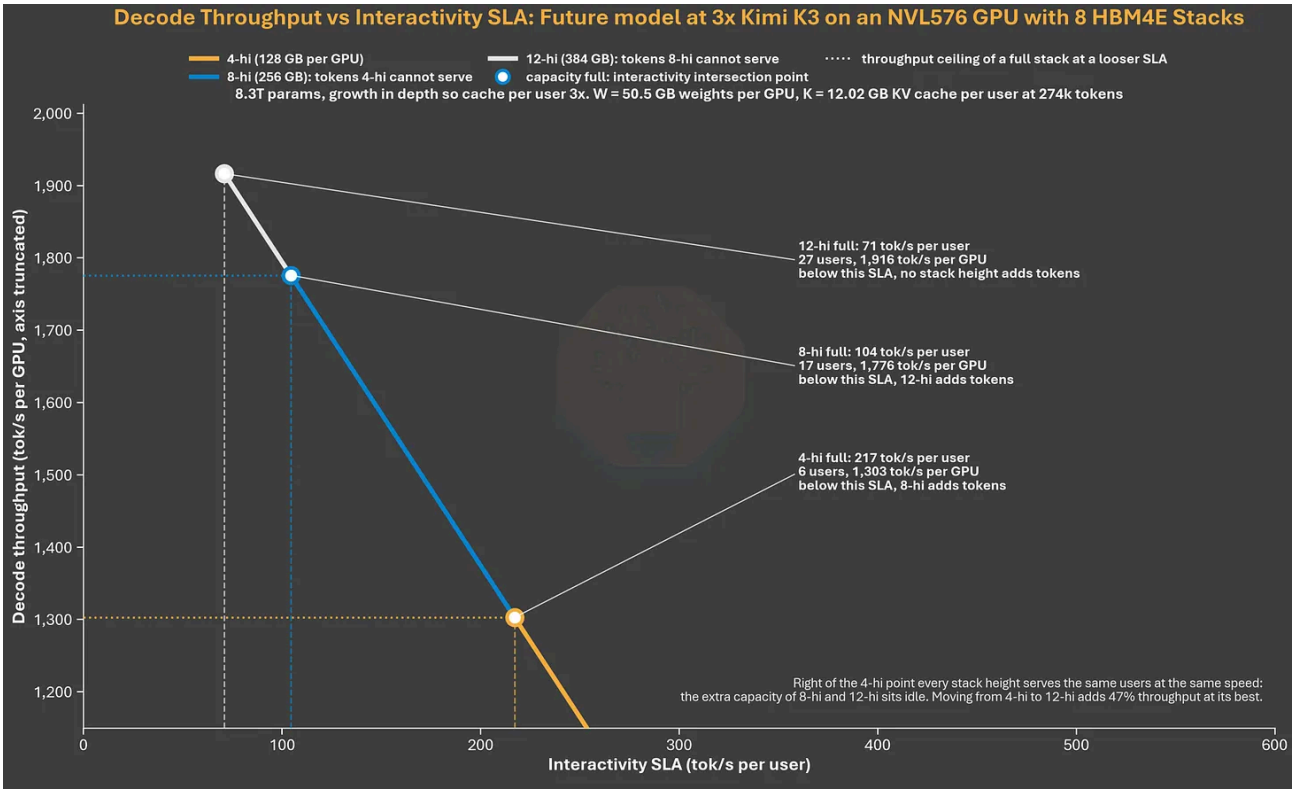
We can see in a real workload that KVCache offload to DDR DRAM can help relieve HBM capacity pressures very effectively. This is for Kimi K3, which doesn't take into account some of the even more extreme cache reduction techniques introduced in the latest DeepSeek v4.1 Flash model, which reduce active KVCache sizes by around 75% compared to v4 Flash. At these reduced KVCache levels, increasing concurrency to utilize

all the HBM capacity is likely to mean that the workload becomes compute and network bound rather than capacity bound, therefore also diminishing the value of a taller HBM stack.

Does 4-hi Give Up Future Proofing Optionality?

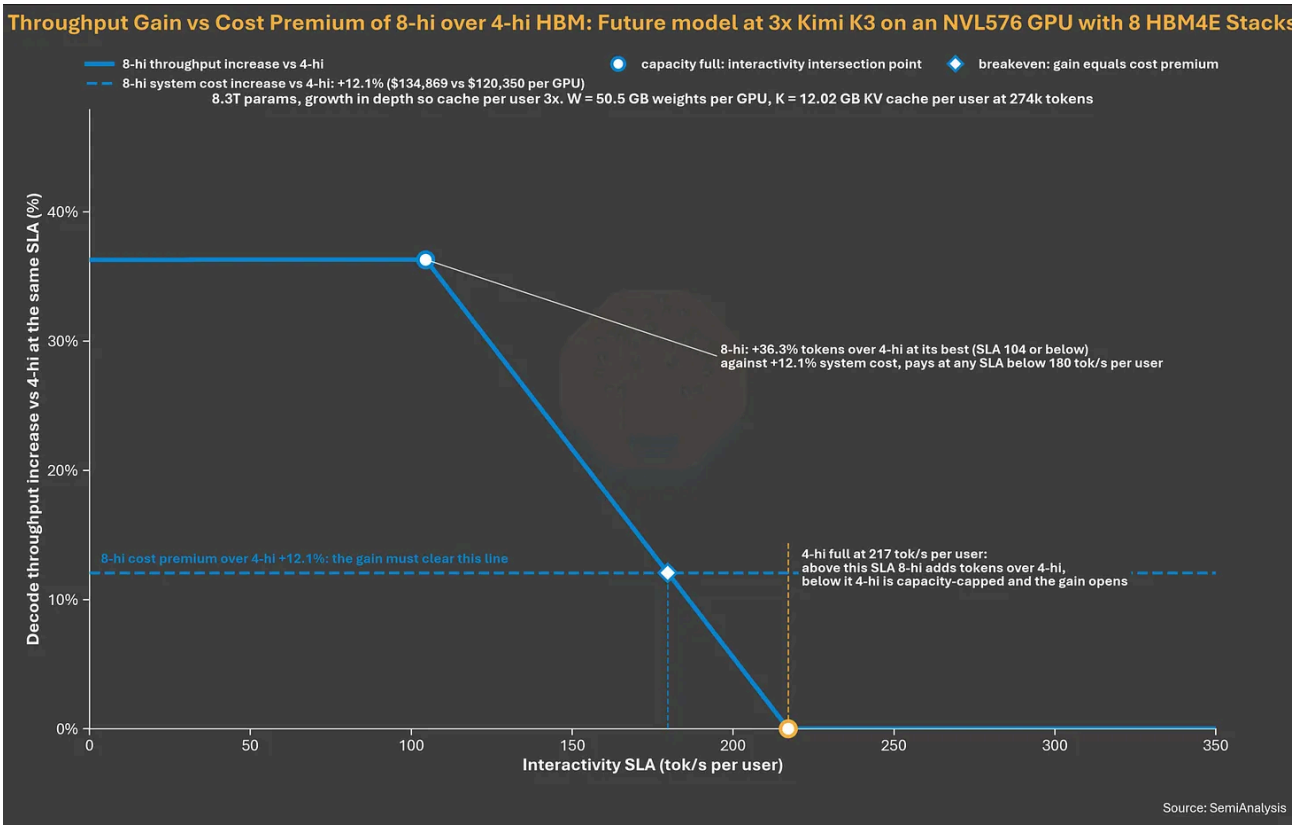
In our K3 roofline analysis using HBM4E on NVL576, the obvious flaw is that we are using a present workload on a future system to show that 4-hi is sufficient. It will only be a year later when these larger scale up domains are available along with HBM4E and frontier models will only have scaled more. In addition, AI servers are deployed for over 5-year lifespans, and will need to be able to continue serving tokens through that period.

The capacity being sacrificed could underestimate optimal capacity needs for future workloads significantly if model sizes for future workloads become much larger. For instance, assuming a future model that is 3x the size of Kimi K3 (along with per user KVCache also tripling in size), the throughput increase from taller stacks of HBM is much better. 12-hi can deliver 47% more total tokens than 4-hi, and 8-hi delivers 36% more tokens, though at half the speed. In this scenario, the additional capacity starts to become more justified and the decision comes down to how valuable the faster tokens are.



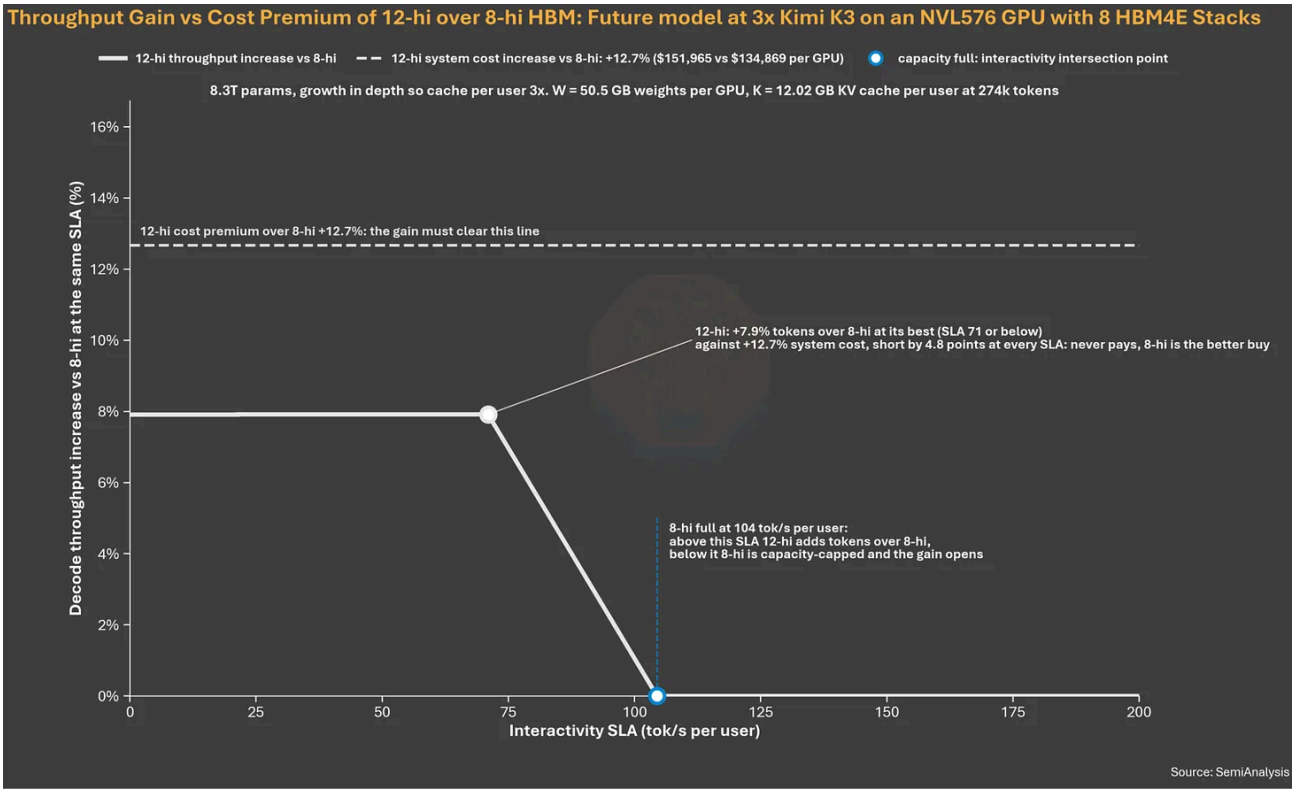
Source: SemiAnalysis

When taking into account costs into this analysis, in this case 8-hi delivers more throughput than the TCO increase at below 180 tok/s/user.



Source: SemiAnalysis

However, going to 12-hi vs 8-hi isn't worth it, nowhere on the curve is 12-hi's increase in token throughput greater than the BOM cost increase.



Source: SemiAnalysis

There are several counter-arguments to this. First, scaling parameters and therefore model weights is no longer as important for model capability. The use of significantly more reinforcement learning as well as more reasoning/thinking time have been key developments that have resulted in more efficient model architecture. Also, it is effectively confirmed that GPT-6 Astra uses looped transformers and there is a lot of speculation that this is the case for Anthropi's largest models as well. Looped transformers effectively run the input through transformer layers multiple times. This scales the model through depth rather than by adding weights.

Second, if 4-hi accelerators become a significant part of the AI server accelerator installed base, researchers will adapt their models so that they can be effectively served with this hardware. Additionally, we understand that it is the Frontier Labs' hardware teams who are the most in favour of 4-hi HBM. These teams are obsessed with hardware/software co-design, and their calling for 4-hi reflects that future models are heading in a direction that don't require that capacity.

Lastly, the big picture is that everyone building and serving models has the goal of serving maximum tokens. Going to 4-hi stacks is what will ensure this…

Maximizing ~~Tokens/Watt~~ Tokens/HBM Wafer

The optimal hardware architecture is what maximizes output per scarce resource. We are in a world of many shortages when it relates to building out AI infrastructure. Power is one of the key ones, which is why optimizing for tokens/W is so important and a key metric that inference providers and hardware architects are trying to optimize for. Of course, money is a finite resource and why performance/TCO has been an evergreen figure of merit driving hardware decisions.

As we mentioned above, one of the other major constraints is the supply of HBM wafers. While not talked about, optimizing for tokens/HBM wafer can be thought of in the same vein as tokens/w and tokens/TCO. The way to do that is simple: use 4-hi HBM. As you can get the same bandwidth from a 4-hi stack as an 8-hi or 12-hi stack, in simple terms the 4-hi means triple the memory bandwidth that can be harvested from each HBM wafer if that wafer was used for 12-hi HBM, or double the bandwidth of an 8-hi wafer. The numbers are even better than that when factoring in the higher packaging yields from 4-hi. As memory bandwidth generally equates to token throughput, this means 4-hi is the way to go in terms of getting the most out of a limited supply of HBM supply.

By going to 4-hi, the amount of HBM cubes harvestable compared to 8-hi could more than double. These cubes are only useful if they can be co-packaged with an accelerator and put into a system. This means that the bottleneck moves away from HBM wafers to other parts of the supply chain like logic wafers (both for the accelerators and the HBM base die), substrates, PCBs, downstream integration capacity as well as many other

components in between. Let's not forget all the power needed to deploy more systems. Overall, though, we believe the rest of the supply chain is able to ramp capacity faster than DRAM supply over the next few years. Even if the AI server supply chain can't match the supply needed to accommodate double the HBM cubes, this also frees wafers for conventional DRAM that can serve other applications that are memory starved because of HBM's cannibalizing effect on overall DRAM supply. The first that comes to mind is server DRAM, with DRAM per socket being cut dramatically to make the most of limited server DRAM supply. This is also very much related to supporting agentic workloads, where large amounts of time are actually spent waiting on a CPU-related execution rather than the GPU to decode tokens.

Behind the paywall, we will discuss what the impact is on memory suppliers and how 4-hi could benefit them, as well as how they are responding. We have significantly more insights, product sku / supplier decisions, and demand / supply details in our Memory Model.

Memory Supplier Impact

While 4-hi has been requested by major customers, so far only Micron is complying and testing it for at least 2 clients. However, Samsung and SK Hynix have pushed back on shipping 4-hi and are not willing to do so yet.

The Korean memory suppliers want to maximize value per HBM stack and bit demand. HBM has been a game-changer for the DRAM profitability as a whole. The huge increase in wafer intensity associated with HBM has consumed a large amount of DRAM wafer starts and has been a key contributor to the current DRAM supply crunch. It is understandable that Samsung and SK hynix want to keep the party going, and downgrading to 4-hi from 8-hi and 12-hi would go in the opposite direction of this.

One way to sweeten the deal for the memory manufacturers would be for 4-Hi to be offered on a premium $/Gb basis versus equivalent 8-hi HBM. If anything, a $/Gb premium may be necessary to amortize the cost of the base die, which becomes a larger % of HBM BOM in a lower stack configuration. The flip side is that there are savings from higher packaging yields from a much simpler and lower stack height assembly.

We believe that memory suppliers can charge a 10% $/GB premium on HBM 4-Hi cubes vs HBM 8-Hi, which is what we have modeled. This means 4-hi would end up being a higher margin product even when taking into account amortizing the cost of the base die. This would result in a win-win for both the suppliers who can enjoy higher margins and customers who benefit from a still attractive $/bandwidth proposition. If Micron ends up being the sole supplier of 4-hi then they can enjoy a premium margin on a product that is more manufacturable. In that case, we don't think Samsung and SK hynix would hold out for too long and they would want a cut of the profits too.

Recommend SemiAnalysis to your readers

Bridging the gap between the world's most important industry, semiconductors, and business.









