The Huawei-DeepSeek Symbiosis

Huawei Ascend 950PR Deep dive & µbenchmarks, DSv4.1 Flash Analysis, DeepSeek support for Ascend, AICPU, Tilelang Ascend

6 HOURS AGO · PAID

147 5 Share

Right before China's national day holidays, DeepSeek released Ascend support for TileLang, DeepGEMM, DeepEP, TileKernels, FlashMLA, and DeepSelect.

Neither can do this alone. Although they have Pangu, Huawei has no frontier lab to prove their chip can train or infer models, DeepSeek has models but cannot buy the compute it needs. Each supplies exactly what the other lacks: Huawei provides the hardware and native software stack; DeepSeek brings the workloads and engineering needed to make that hardware useful for its models. September 30's release of Ascend support in DeepSeek repositories is an artifact of this trade, where the exchange runs through the kernel-authoring language.



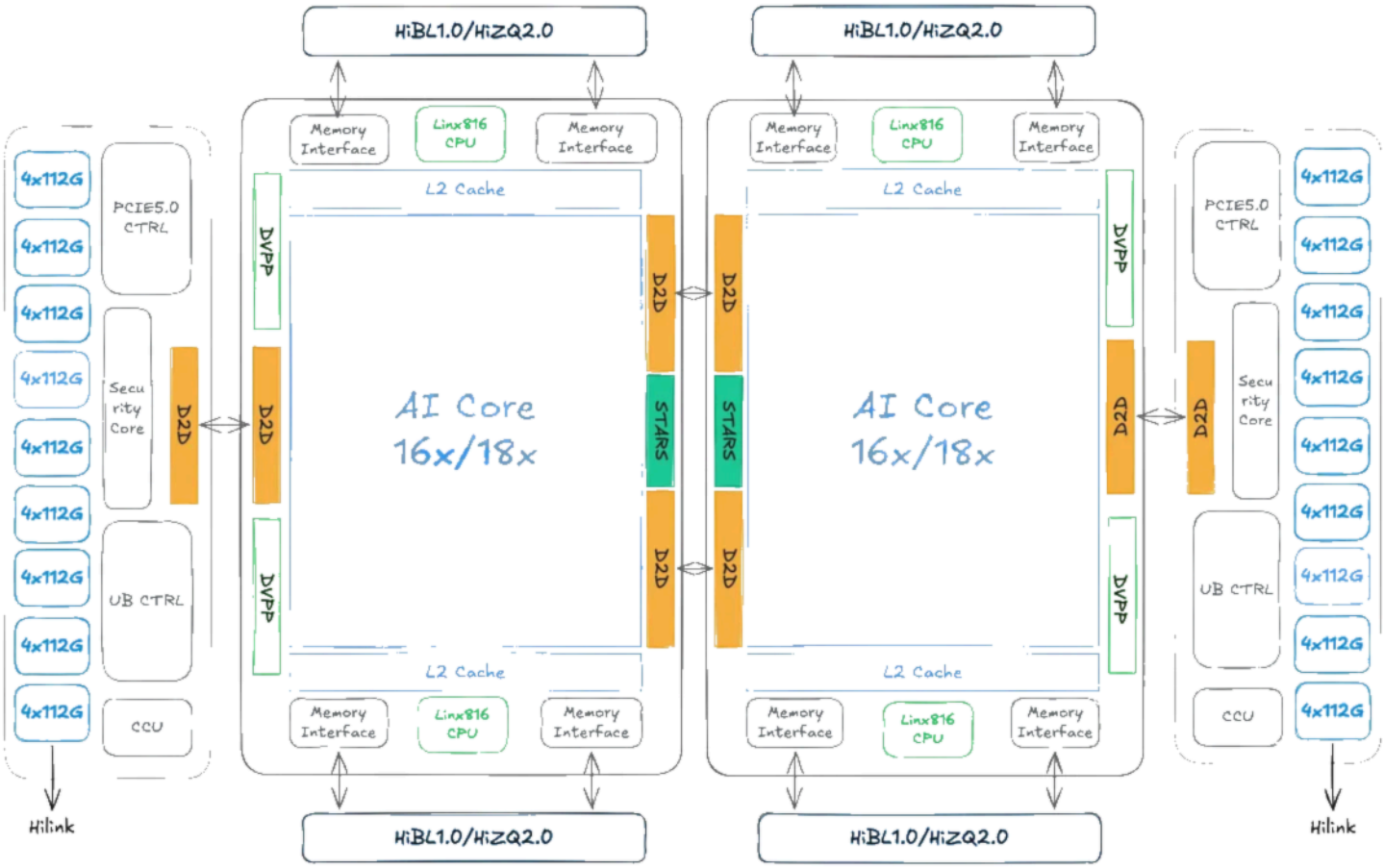
Source: SemiAnalysis - *Note that the diagram is a simulation.*

In 2028 or even 2027, DeepSeek will likely be primarily using Huawei's stack, especially with the external and internal pressure to build a fully domestic chinese ecosystem. As we show here and in future articles, the Huawei ecosystem is already mature and well developed, with strong foundations from the Ascend time and DeepSeek engineers building further ontop.

In this article, we'll first examine Ascend 950PR's architecture and benchmark its compute and memory performance. We then explore DeepSeek's software releases, from TileLang's compiler capabilities to hardware-specific kernel optimizations and rack-scale communication. Finally, we analyze DeepSeek-V4.1-Flash's execution profile.

Ascend 950

We deep-dived into CANN and the Ascend 950 architecture in our [DeepSeek V4 Pro article](). Since then, more interesting details have been revealed.



Source: SemiAnalysis

Starting from the chip, many variants of 950PR/DT have appeared with varying core counts, HBM stack height, and cache sizes. Some are likely either virtualised/simulated, or for testing only. This is because only a fraction of dies and HBM stacks meet the target spec, due to both outright defects and parametric shortfalls in power and performance, so Huawei bins both compute and memory into multiple SKUs and pairs weaker memory with weaker compute, rather than binning only compute as Nvidia does.

950DT — SKU Comparison

SKU	Chip	Core Count				Frequency		Memory	L2 Cache		DDR Rate		Virtualisation Options
		AI Core	Cube Core	Vector Core	AI CPU Core	Cube Core	Vector Core	(GiB)	(MiB)	Base	Read	Write	
DT_9591	950DT	36	36	72	6	1800	1800	144	128	40	40	40	4,8,16,32
DT_9592	950DT	36	36	72	6	1800	1800	96	128	40	40	40	4,8,16,32
DT_9595	950DT	36	36	72	6	1800	1800	96	128	31	31	31	4,8,16,32
DT_9596	950DT	36	36	72	6	1800	1800	64	128	31	31	31	4,8,16,32
DT_95A1	950DT	36	36	72	6	1800	1800	144	128	40	40	40	4,8,16,32
DT_95A2	950DT	36	36	72	6	1800	1800	96	128	40	40	40	4,8,16,32
DT_9581	950DT	32	32	64	6	1650	1650	144	128	40	40	40	4,8,16,32
DT_9582	950DT	32	32	64	6	1650	1650	96	128	40	40	40	4,8,16,32
DT_9582x	950DT	32	32	64	6	1650	1650	96	128	40	40	40	4,8,16,32
DT_9583	950DT	32	32	64	7	1650	1650	126*	112	35	35	35	4,8,16,32
DT_9584	950DT	32	32	64	7	1650	1650	84	112	35	35	35	4,8,16,32
DT_9585	950DT	32	32	64	6	1650	1650	96	128	31	31	31	4,8,16,32
DT_9586	950DT	32	32	64	6	1650	1650	64	128	31	31	31	4,8,16,32
DT_9587	950DT	32	32	64	7	1650	1650	84	112	27	27	27	4,8,16,32
DT_9588	950DT	32	32	64	7	1650	1650	56	112	27	27	27	4,8,16,32
DT_9571	950DT	28	28	56	7	1650	1650	144	128	40	40	40	7,14,28
DT_9572	950DT	28	28	56	7	1650	1650	96	128	40	40	40	7,14,28
DT_9573	950DT	28	28	56	5	1650	1650	126*	112	35	35	35	7,14,28
DT_9574	950DT	28	28	56	5	1650	1650	84	112	35	35	35	7,14,28
DT_9575	950DT	28	28	56	7	1650	1650	96	128	31	31	31	7,14,28
DT_9576	950DT	28	28	56	7	1650	1650	64	128	31	31	31	7,14,28
DT_9577	950DT	28	28	56	5	1650	1650	84	112	27	27	27	7,14,28
DT_9578	950DT	28	28	56	5	1650	1650	56	112	27	27	27	7,14,28
DT_950x	950DT	8	8	16	5	1650	1650	24	32	10	10	10	2,4,6
DT_950y	950DT	8	8	16	5	1650	1650	16	32	7	7	7	2,4,6

950PR — SKU Comparison

SKU	Chip	Core Count				Frequency		Memory	L2 Cache		DDR Rate		Virtualisation Options
		AI Core	Cube Core	Vector Core	AI CPU Core	Cube Core	Vector Core	(GiB)	(MiB)	Base	Read	Write	
PR_9599	950PR	36	36	72	6	1800	1800	128	128	16	16	16	4,8,16,32
PR_9589	950PR	32	32	64	6	1650	1650	128	128	16	16	16	4,8,16,32
PR_958b	950PR	32	32	64	7	1650	1650	112	112	14	14	14	4,8,16,32
PR_9579	950PR	28	28	56	7	1650	1650	128	128	16	16	16	4,8,16,32
PR_957b	950PR	28	28	56	7	1650	1650	112	112	14	14	14	7,14,28
PR_957bx	950PR	28	28	56	7	1650	1650	112	112	14	14	14	7,14,28
PR_957c	950PR	28	28	56	5	1650	1650	84	112	14	14	14	7,14,28
PR_957d	950PR	28	28	56	5	1650	1650	96	96	12	12	12	7,14,28
PR_950z	950PR	4	4	8	5	1650	1650	24	16	2	2	2	2,4,6

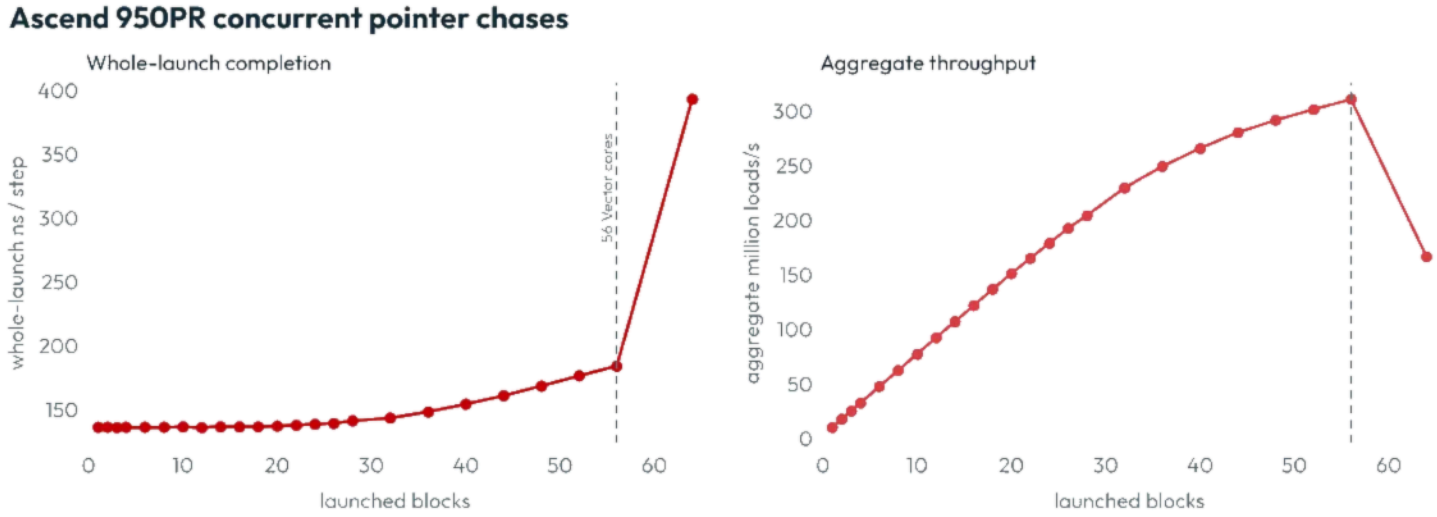
Source: SemiAnalysis

This isn't new, 910B also has variants such as 910B1, 910B2, 910B3, 910B4. For the following microbenchmarks, we use an Ascend 950PR_9579, with 28 cube cores and 128 GiB HBM.

```
root@ca3bb851f93f:/workspace# npu-smi info
+------------------------------------------------------------------------------------------+
| npu-smi 25.7.rc1.6                              Version: 25.7.rc1.6                      |
+--------+-----------------+--------------+-----------------------------------------------+
| NPU ID | Name            | Health       | Power(W)    Temp(C)           Hugepages-Usage(
|        |                 | Bus-Id       | NPU Util(%) Memory-Usage(MB)  HBM-Usage(MB)
+--------+-----------------+--------------+-----------------------------------------------+
| 0      | Ascend950PR     | OK           | 203.3       56                0     / 0
|        |                 | 0000:12:00.0 | 0           0    / 0          5242  / 131072
```

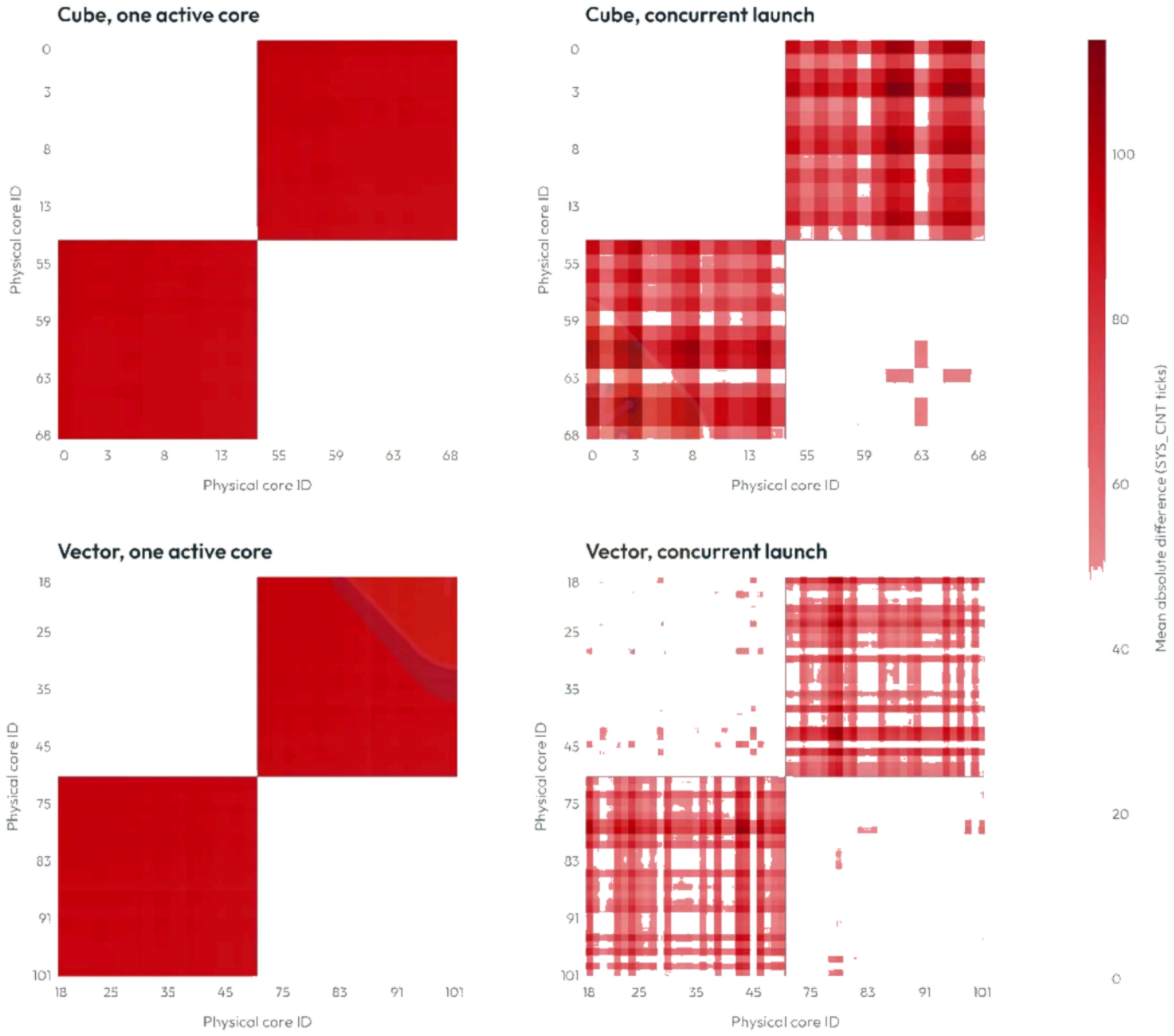
```
+--------+------------------+-------------+---------------------------------------+
+---------------------------+-------------+---------------------------------------+
  NPU ID                    | Process id  | Process name                | Process memory(MB)
+---------------------------+-------------+---------------------------------------+
  No running processes found in NPU 0
+---------------------------+-------------+---------------------------------------+
```

Running point chases reading 16 KiB of integer payload per launch, it stays near 134ns through 18 blocks, rises to 182ns at 56, then jumps to 391ns at 64. This is consistent with a second wave beyond 56 Vector cores.



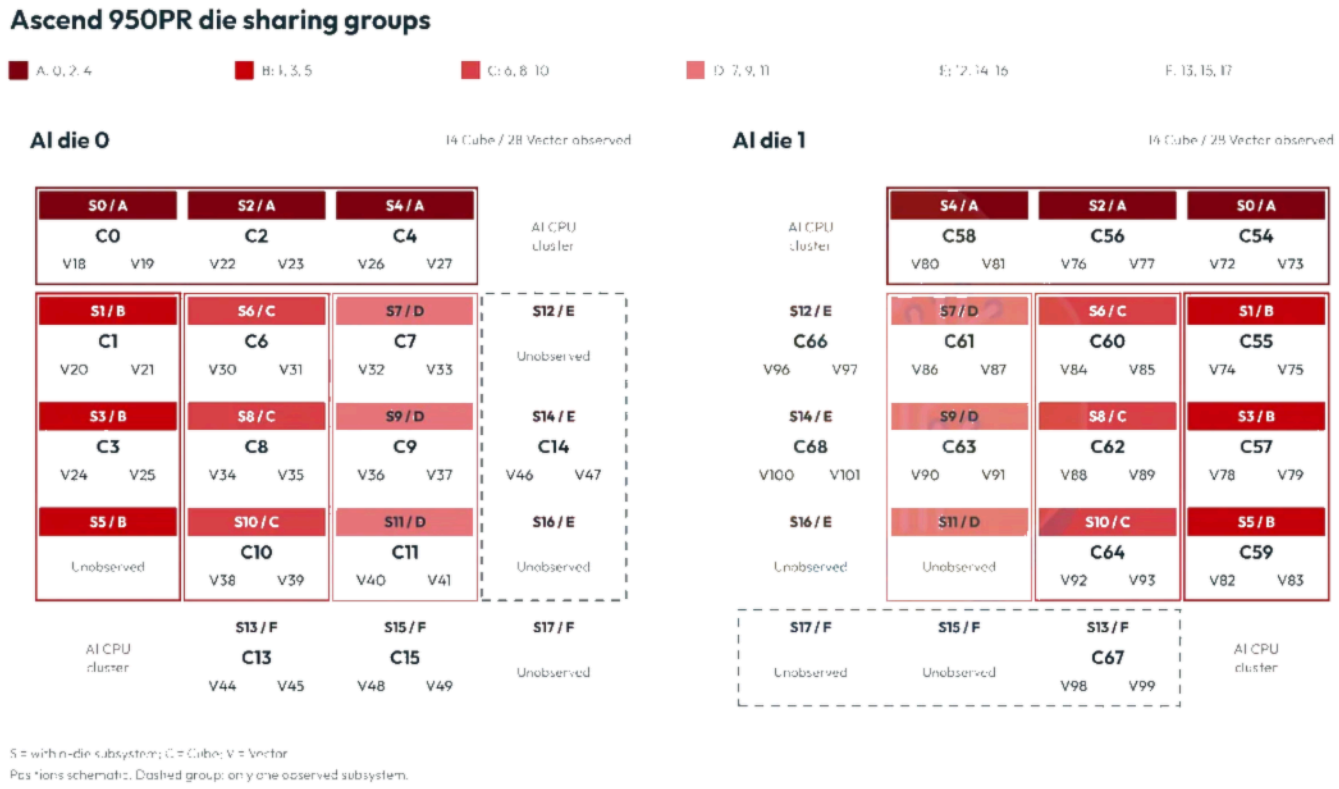
Source: SemiAnalysis

Then by running a scalar pointer-chase probe separately on every Cube and Vector core, recording load timings at matching addresses throughout a 128 MiB chain, we get the following graph. Each cell's color represents the mean absolute difference between two cores' address profiles, producing two groups that match the decoded dies.



Source: SemiAnalysis

Going further, by timing a Vector core's pointer chase while another core runs either memory traffic or register-only work; the extra slowdown reveals which subsystems share resources, supporting groups such as {0,2,4} and {1,3,5}. Mixed Cube/Vector launches identify each subsystem's one Cube and two Vectors, which we draw together within those groups, with physical positions remaining schematic.



Source: SemiAnalysis

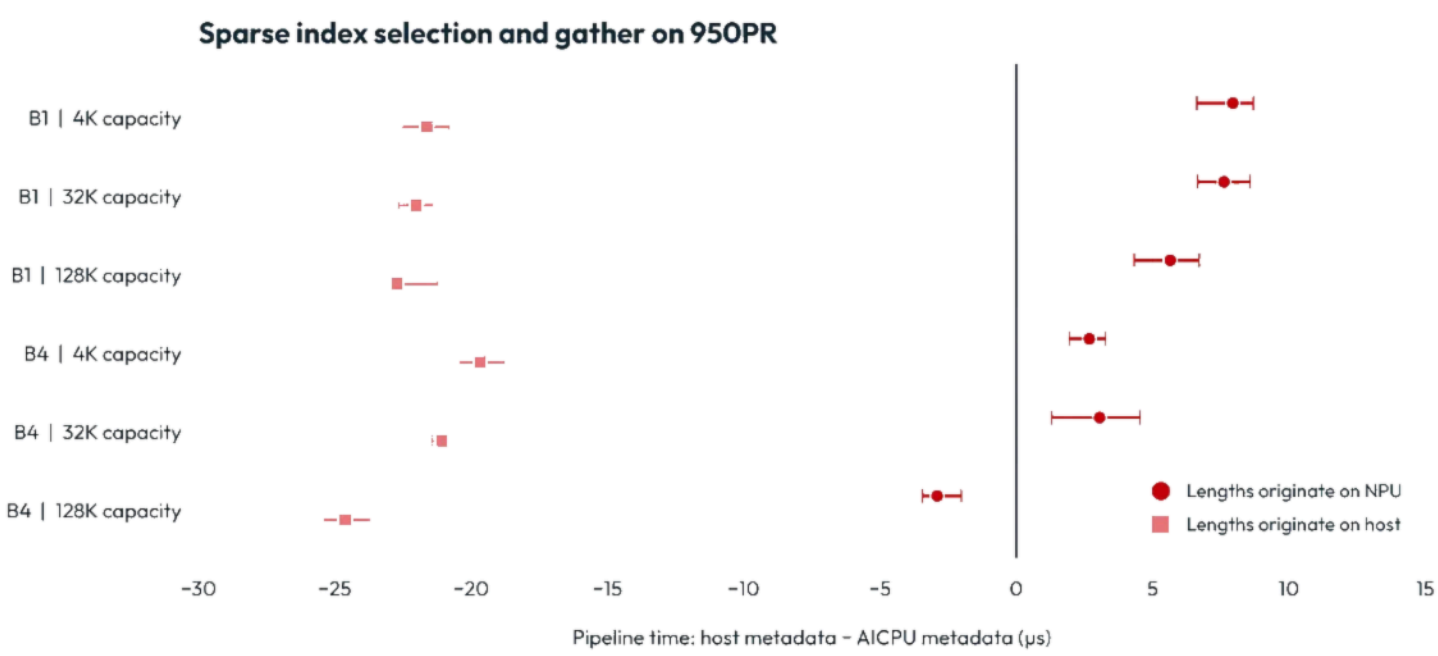
CPU within the NPU

Each 950PR comes with a variable number of Linx816 AArch64 ELF AICPUs. The MIDR numbers imply these cores are likely the same as the middle cores in the Kirin 9000S/9010. These AICPU cores have instruction set:

```
fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp cpuid
asimdrdm jscvt fcma lrcpc dcpop sha3 sm3 sm4 asimddp sha512 sve
asimdfhm dit uscat ilrcpc flagm sscs sb paca pacg dcpodp flagm2
frint svei8mm svebf16 i8mm bf16 dgh rng ecv
```

For this SKU, there are 7 configured, 6 visible, and 5 executing AICPU cores. Other than device control it can provide CPU-operator support, such as KV metadata. The AICPU helps out for short work where D2H overhead take up most of the time.

For DSA, a small scheduling step reads the current sequence lengths and assigns indexer work to accelerator cores. Cube/Vector cores then compute INT8 indexer scores and select the top 2,048 indices. These planning, index selection, and gather steps are ran on AICPUs in vLLM-ascend implementation, and we compare their performance to running on the host CPU.



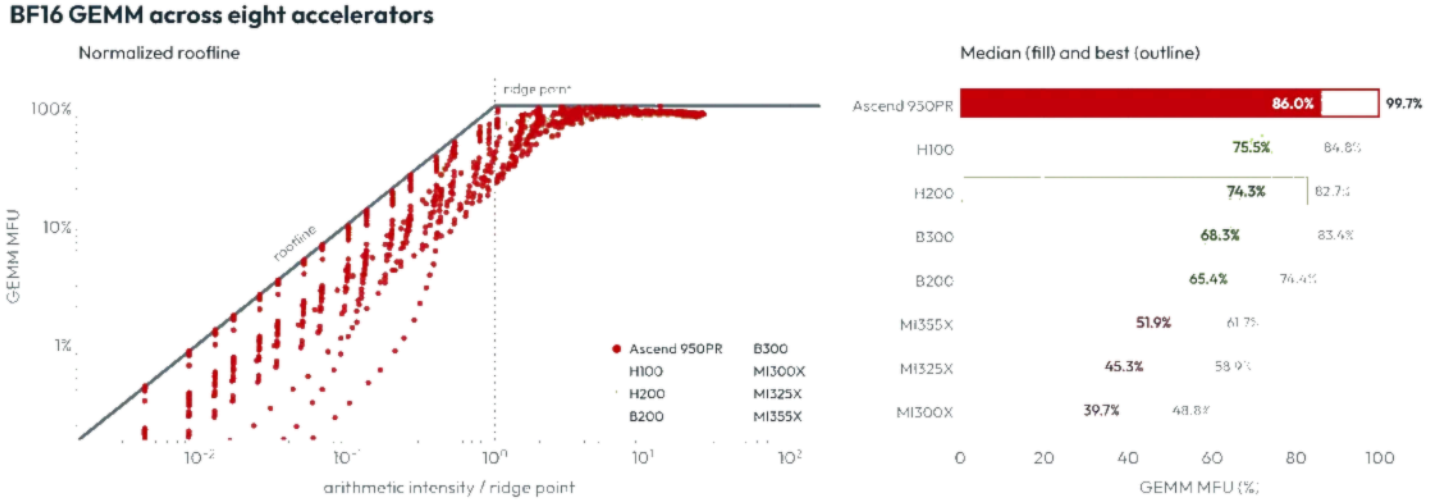
Source: SemiAnalysis

Due to their limited compute performance, AICPU is only good for short work where overhead dominates, and work where data starts and is consumed at the NPU.

Microbenchmarks

Ascend 950PR is designed for prefill and recommendation. It gets lower memory bandwidth and core than its bigger brother the 950DT.

The results below are not from DeepSeek's contributions but from stock `torch.matmul` on ascend resolving to `aclnnMatmul_MatMulV3Common_MatMulV3` . We compared 1,055 BF16 GEMM shapes from several open-sourced models. Huawei's best kernels hits 99.7% of the ceiling, showing DeepSeek's 99+% from DeepGEMM isn't the only way to maximize utilization. Huawei engineers did good work too.

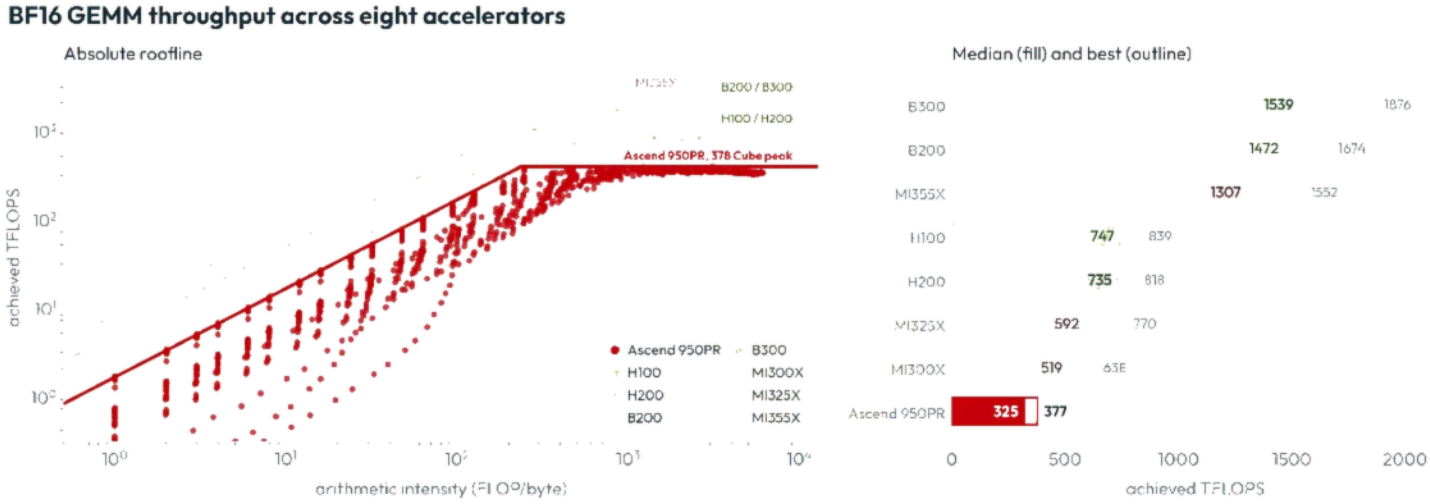


Source: SemiAnalysis

The advantage is concentrated in compute-bound work. On the shapes below Ascend's ridge, its median achieved/attainable fraction is 41.5% versus H100's 46.8%. Some of Nvidia's results were power-limited and did not hit maximum SM clock: median power around 690-698 W out of a 700 W limit, and median sampled SM clocks at 1335-1470 MHz, below the max of 1830 MHz. Huawei approximated 599-600 W, at 1500-1550 MHz against its 1650 MHz reference.

Ascend's MXU design may contribute to some of this high utilization, Ascend stages matrix operands in dedicated L0A/L0B buffers and accumulators in L0C, with overlapping movement, computation and result handling. Huawei associates the enlarged L0C with more flexible tiling and reuse.

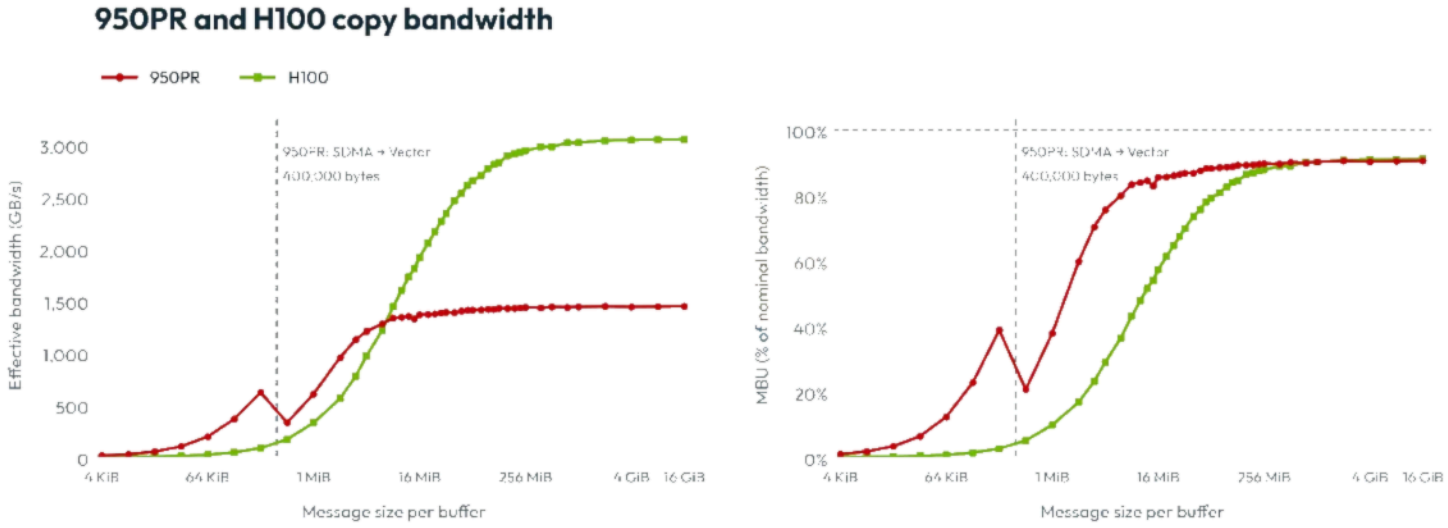
But of course, 950PR's falls behind on absolute values, sitting somewhere between A100 and MI300X.



Source: SemiAnalysis

MemBW

Measuring HBM memory bandwidth, we see 950PR's effective bandwidth and bandwidth utilization surpassing H100 at small message size per buffers, and nearly matching H100 in bandwidth utilization at large message sizes.



Source:

A dip appears in the curve as Ascend switches copy methods to a Vector-core TensorMove kernel that has more startup cost, so a slightly larger copy suddenly takes longer. For very large copies, that extra cost becomes relatively small, so bandwidth rises again.

Note that high FLOPs utilization and bandwidth utilization does not necessarily translate to high MFU/MBU. Training, especially with FSDP, requires collectives, checkpointing, and data moving that can be hard to overlap.

As we mentioned in our Jalepeno article, a “chip” in benchmarks is just a social construct. Multiple dies can be glue to form a bigger “chip”. Thus performance per watt or TCO is the best comparison. But for China, watts are far cheaper. We’ll analyze Ascend’s perf per TCO in a future article.



BRYAN SHAN, MYRON XIE, AND 4 OTHERS · 8月25日

Read full story →

Hardware raises the ceiling, but software lets you reach that ceiling. Although Huawei’s hardware may be beyond ~5 years behind(compared to A100), the software is right at the frontier.

DeepSeek Open Source

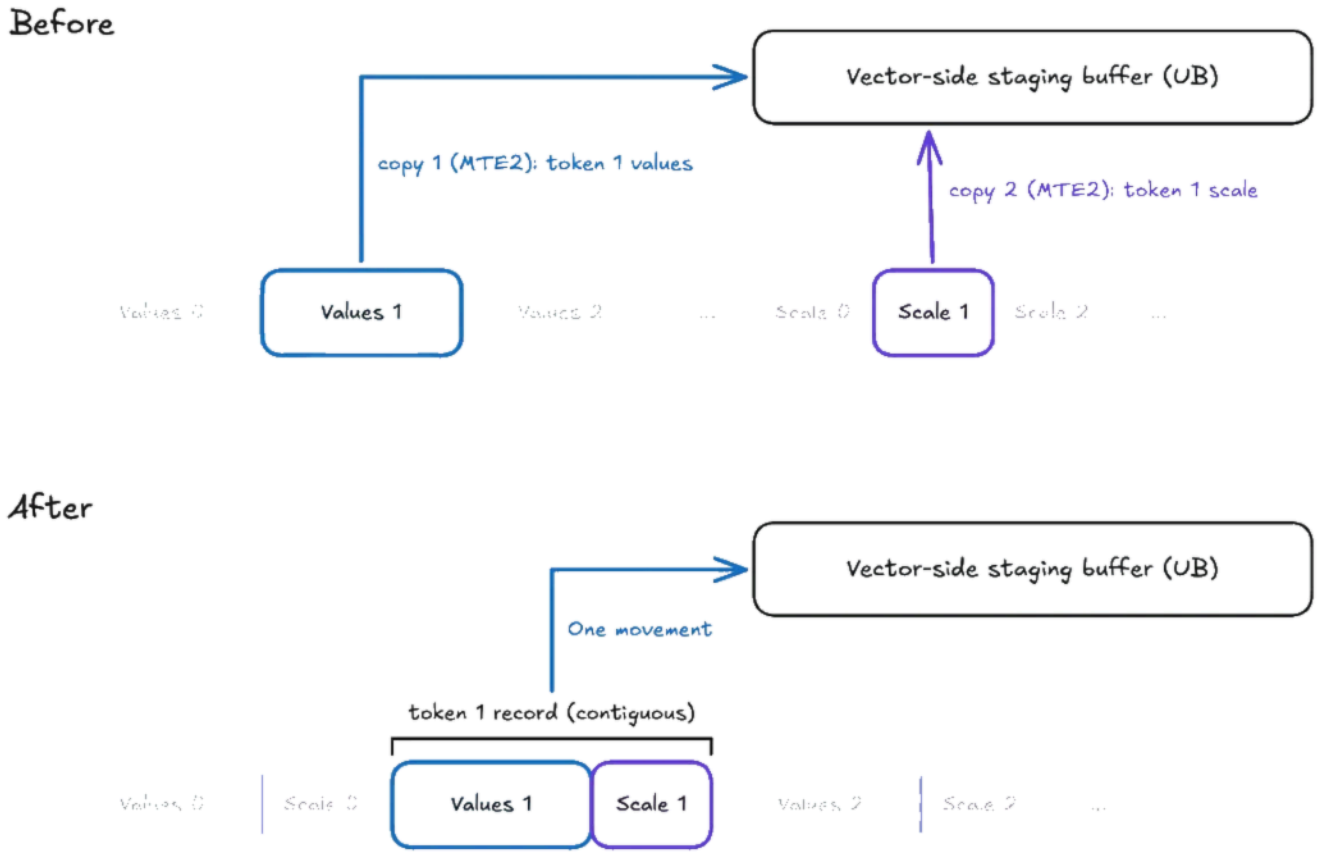
For the 6 repos that received ascend support(TileLang, DeepGEMM, DeepEP, TileKernels, FlashMLA, DeepSelect), only TileLang is usable as of writing, and CANN 9.2.0 has not been released. Deepseek requires CANN 9.2.0 which was released after

time of writing, and a C++20 compiler is required which CANN 9.2.0-beta.2 does not provide. The HDK will be released on Oct 15. This comes after DeepSeek's open source week in 2025, where some of these repositories were released with Nvidia support only.

From the results above, we can see Huawei can already execute operations well; the symbiosis is about making the rest of the application & the work required to support it more efficient and reusable.

Engineering runs both ways

Huawei's architecture is already influencing how DeepSeek's software organizes data. Earlier FlashMLA KV-cache layouts stored quantized values and their scaling factors in separate regions. On Ascend, fetching a token then required separate transfers for its values and scales. The new implementation places them together, reducing the number of transfer requests by the memory transfer engine(MTE).



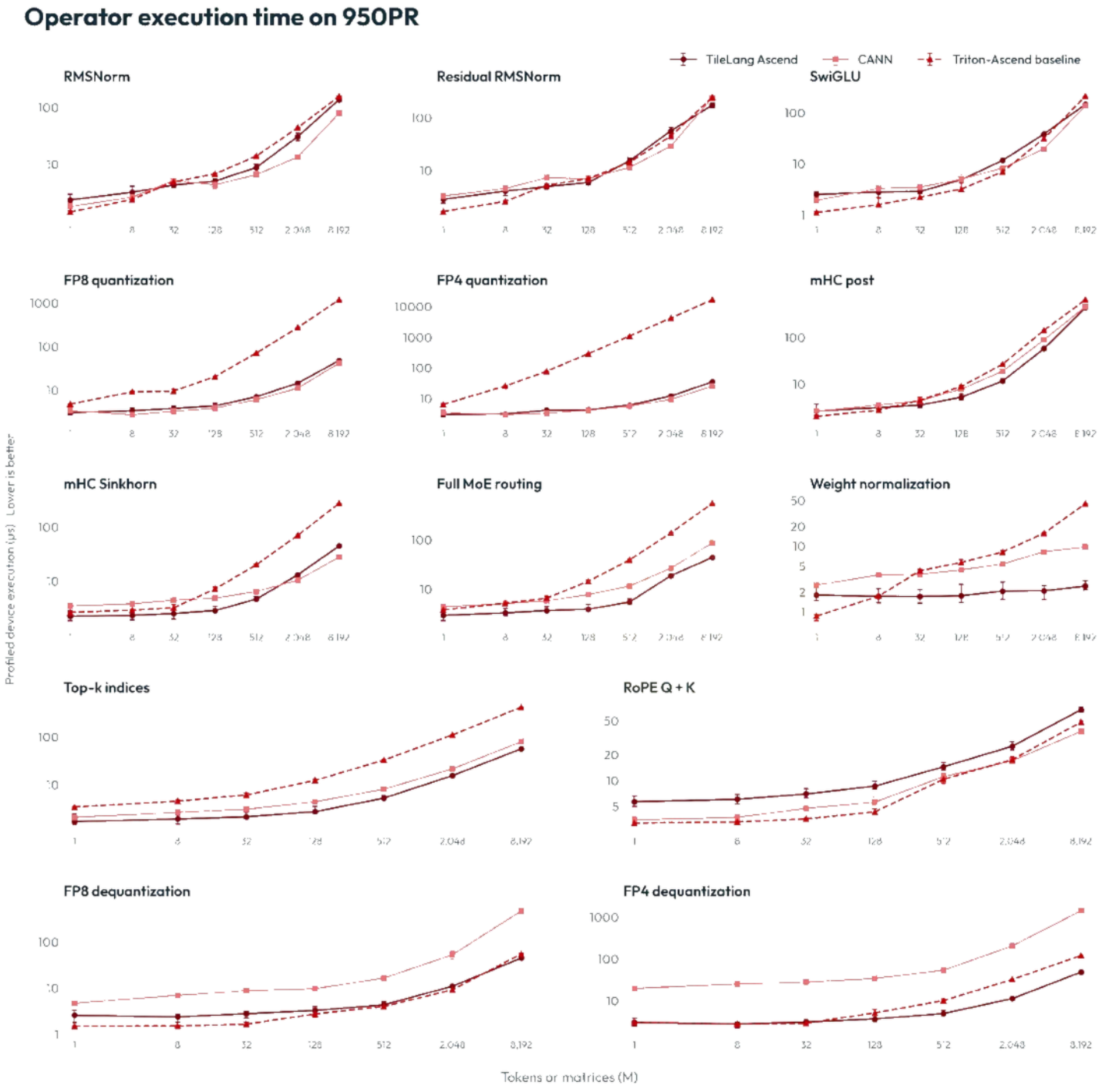
Source: SemiAnalysis

From fast kernels to reusable kernel engineering

Ascend 950 backend is merged into the main TileLang repository, rather than existing only in a separate Huawei adapter. It adds native compilation, scheduling, synchronization and runtime integration for Ascend's execution model.

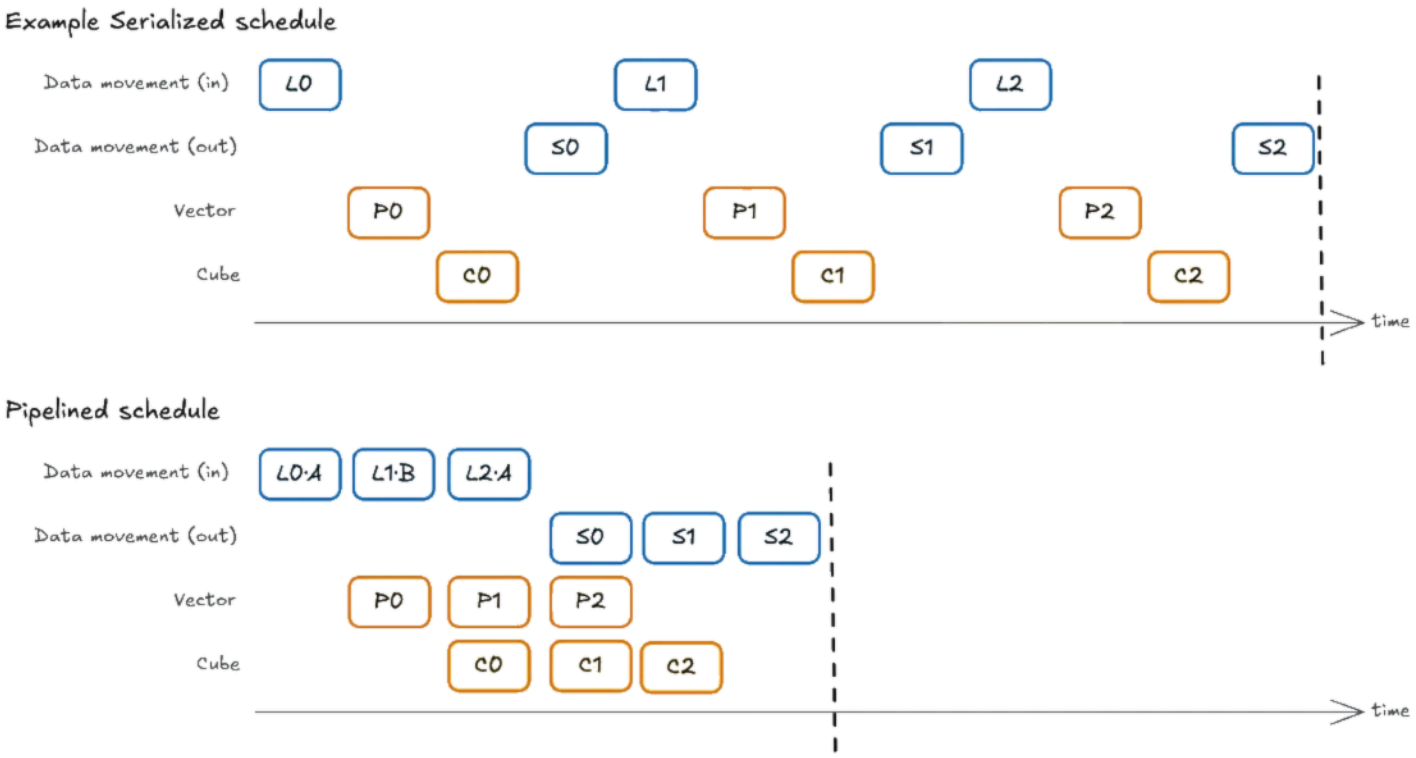
TileLang makes Ascend kernels easier to write, but developers still need to account for the hardware. They choose the calculations, tile sizes and where data is stored. The compiler handles much of the coordination between Cube cores, Vector cores and data transfers.

Digging below the implementation and looking at its performance against CANN and Triton-ascend, it can match and beat CANN at certain cases; the preferred implementation changes with size. This is a warm-cache simple and naive benchmark.



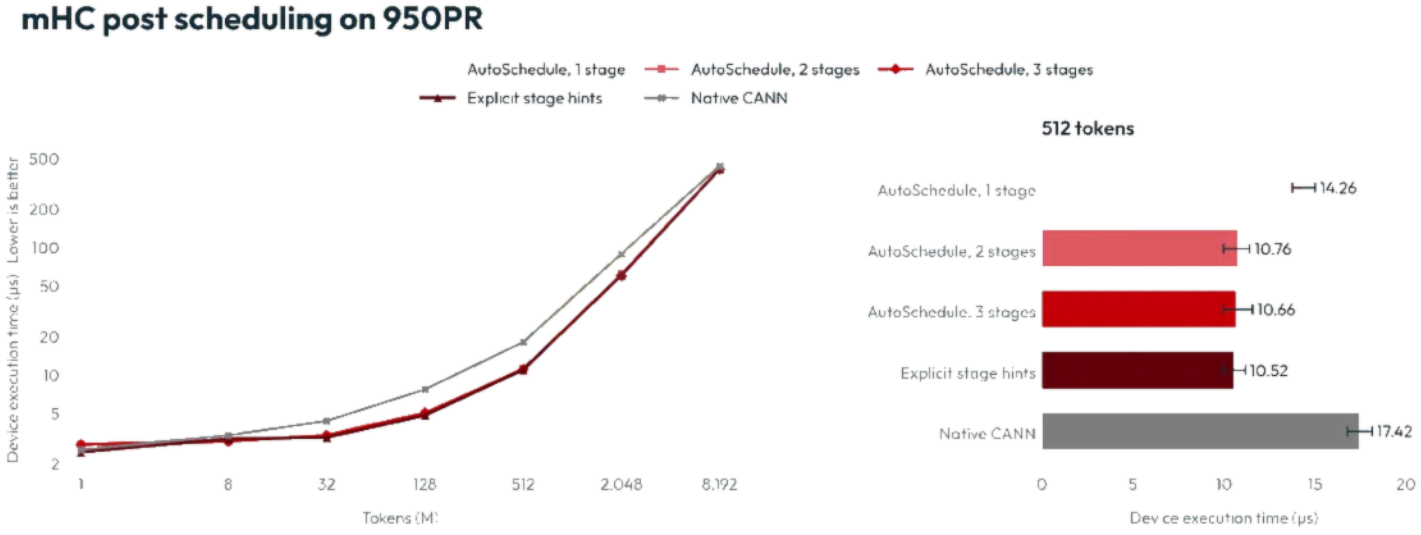
Source: SemiAnalysis

AutoSchedule uses data dependencies and estimated execution times to plan which operations can overlap. It also chooses how many buffer slots to use within the available on-chip memory, allowing upcoming data to load while the current tile is being processed. Developers can guide these decisions when the default schedule is not suitable.



Source: SemiAnalysis

On 950PR, AutoSchedule helps especially for smaller sizes, reducing execution time by up to 40% for a single operation.

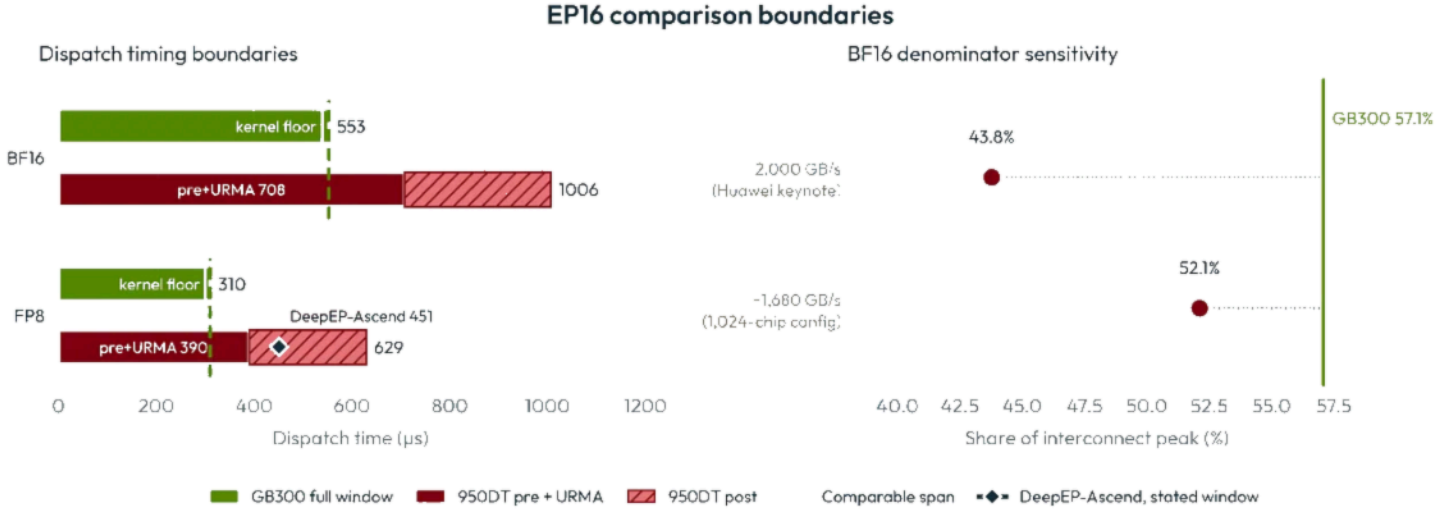


Source: SemiAnalysis

The compiler also inserts synchronization so operations wait for the data they need, while removing waits that other dependencies make unnecessary. It checks conditional branches too, so it does not assume an operation has completed when that branch might never execute. This reduces the synchronization work programmers must manage themselves.

DeepEP

Huawei has joined NVIDIA in commercializing large scale-up AI systems, with its Ascend supernodes extending tightly coupled accelerator interconnects across multiple racks from the 910 CloudMatrix384 to the 950 SuperPod,enabling wideEP scale-out. In the 30th September release, Huawei also released EP16 dispatch/combine benchmarks over URMA in the 950DT superpod.



Source: , , SemiAnalysis

Huawei's dispatch is three sequential operators (pre/URMA/post) whereas GB300 is one fused call, and their post work includes receive completion, which our measurement includes, but also includes expansion by expert which GB300's results don't.

DeepEP-Ascend is a port of DeepEP V2.5, with the same python API but calls URMA and HCCL underneath. In July 2025, Huawei's Ascend team proposed DeepEP-Ascend under `deepseek-ai`, but a DeepEP maintainer declined. Today, DeepEP-Ascend is distributed under DeepSeek's own organization.

Ascend's separate Cube and Vector execution paths create opportunities to overlap communication and computation, but do not make that overlap free. Expert

computation can also require vector processing and data movement, while communication competes for shared memory bandwidth. Therefore there's a tradeoff on how much communication can be hidden without slowing the computation that runs alongside.

MoE also uses limit_core_num(32, 1) for router GEMM and shared Gate/Up MatMul. For this optimized Decode shape, the default MatMul uses vec acceleration; by controlling it to use the Cube path, reducing the footprint of vec, and freeing up vec resources for parallel quantification, activation, and communication tasks.

MoE 还对 Router GEMM 和 Shared Gate/Up MatMul 使用 `limit_core_num(32, 1)`。针对本次优化的 Decode shape，默认 MatMul 会使用 vec 加速；通过控核使其采用 Cube 路径，减少对 vec 的占用，为并行的量化、激活和通信任务释放 vec 资源。这一调整与跨流等待配合，避免算子各自优化后反而争抢资源。 This change needs to work together with cross-stream waiting, so that operators optimized independently don't end up competing for the same resources.



Source:

This resource-balancing problem is not unique to DeepSeek. Huawei's Kimi K3 recipe explicitly restricts vector use to leave capacity for concurrent work, and chooses between dedicated CCU and Vector-based communication according to what must overlap. Faster individual operations are not enough: their resource demands must also fit together.

DeepEP-Ascend's documentation identifies contention between combine processing and URMA communication for HBM resources, and says Huawei firmware improvements are planned to reduce it. This is the practical form of co-design: the model exposes a bottleneck, the implementation adapts, and the platform's development priorities respond to what remains.

All planned

In May's unverified DeepSeek investor meeting transcript, Liang Wenfeng said:

> "So, when it comes to the chip gap between China and the US, I think the software ecosystem gap will eventually disappear. But on the hardware side, we face two separate disadvantages: it takes four Huawei accelerator cards to match one

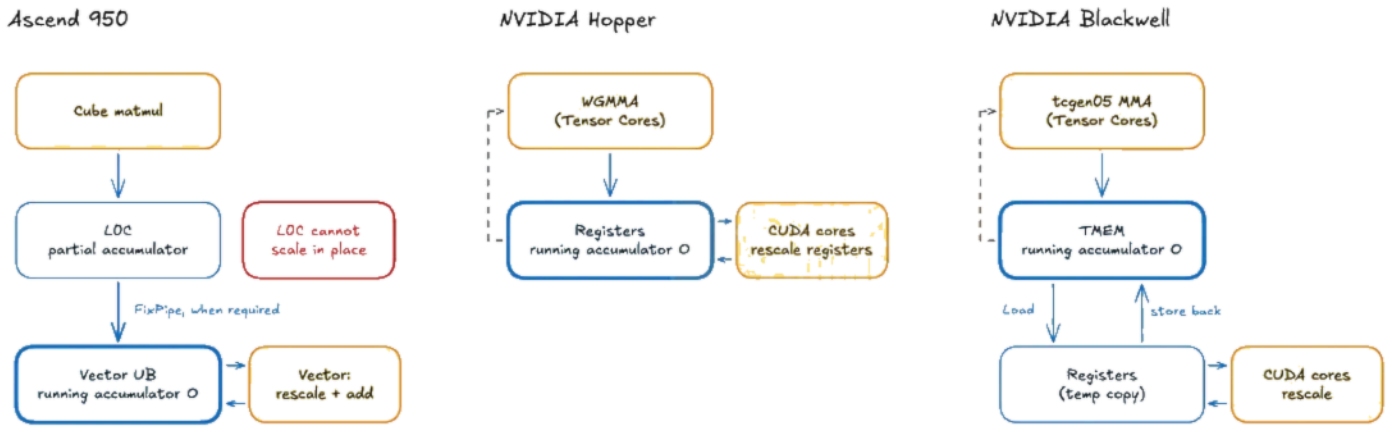
> NVIDIA card, and Huawei is also roughly two years behind in bringing its hardware to market."

4 months later, the ecosystem gap did indeed narrow. The same transcript mentioned TileLang and helping Huawei get the ecosystem right.

DSv4.1 Flash Optimizations

Another part of the DeepSeek-Huawei symbiosis is that DeepSeek provides Huawei with weights, supporting Day-0 optimized releases, such as DeepSeek V4 and V4.1.

Ascend's accumulator design forces a different approach to attention. The Cube's L0C buffer cannot rescale accumulated outputs in place, while returning a rescaled output to L0C would require an expensive identity-matrix multiplication. DeepSeek instead keeps the running output accumulator in Vector UB, transferring partial sums from L0C when needed for rescaling and accumulation. Hopper holds its accumulator in registers, while Blackwell provides loads and stores between tensor memory and registers, avoiding this particular restriction



Source: SemiAnalysis

Profile

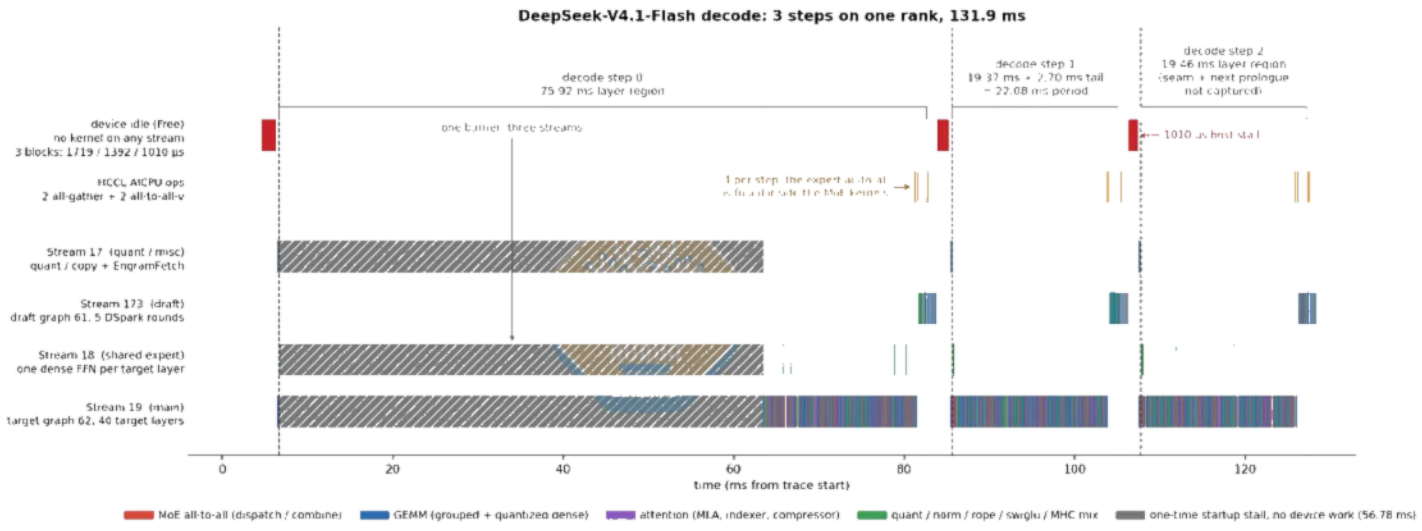
Huawei also released a profile of one rank of DeepSeek-V4.1-Flash on a Huawei Ascend 950DT deployment of 32 chips, together with its host threads. In June, we covered a similar profile, that of DeepSeek V4 Pro's.



BRYAN SHAN, CAM QUILICI, AND 5 OTHERS · 6月9日

Read full story →

Many components are similar, like the use of AICPU and separation of some layers to different streams.

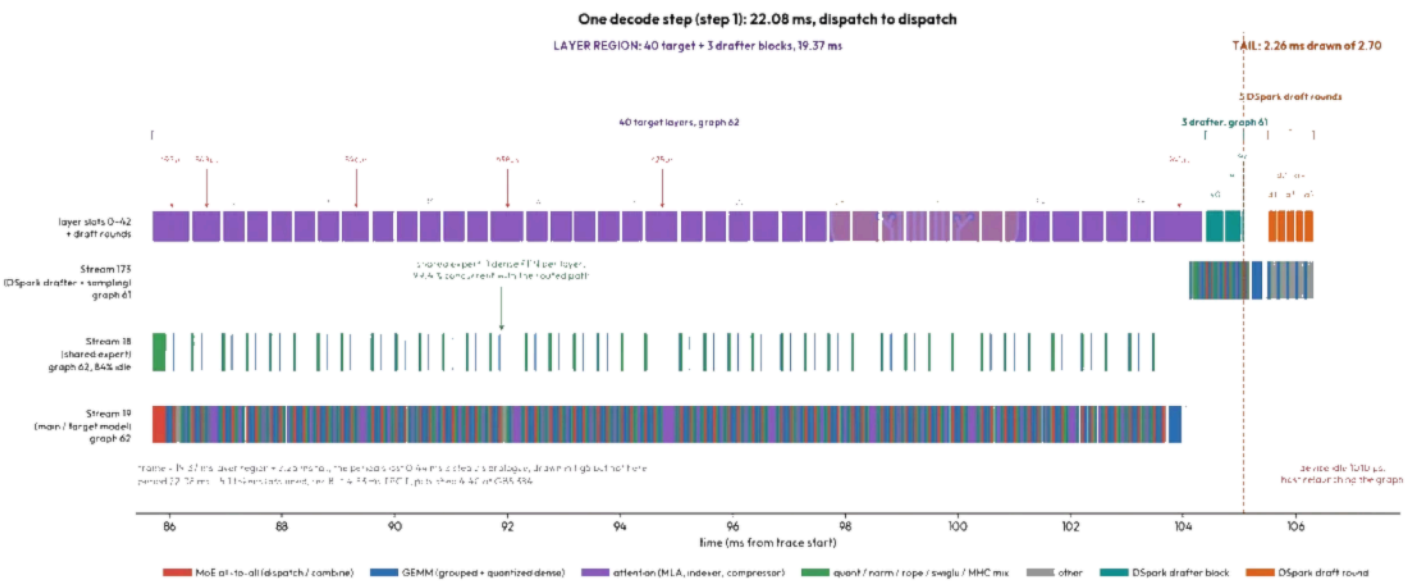


Source: SemiAnalysis

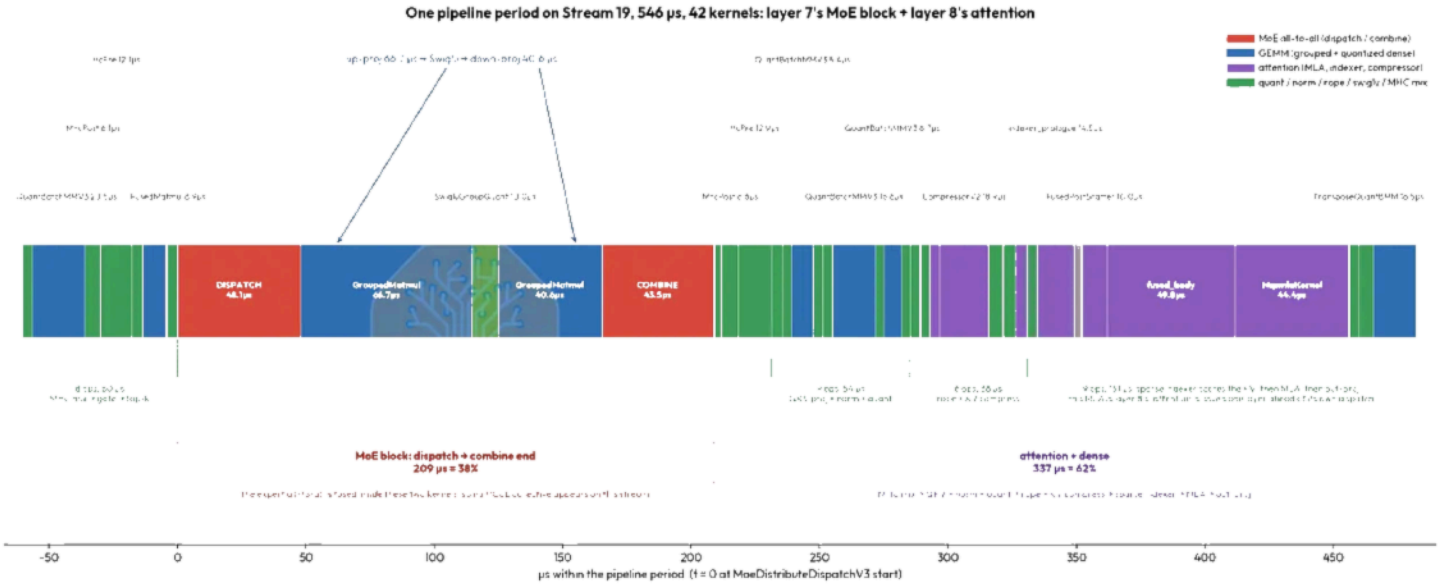
The execution order is 40 target-model layers, the target LM head, 3 DSpark drafter blocks, the draft LM head, and 5 sampling rounds. The main compute runs on Stream 19; the drafter's run on Stream 173. The V2 MC^2 kernels that fused computation and communication from the DSv4 Pro trace shows up again, as V3 this time. This is why only 4 AI CPU-launched HCCL operations appear per step, against 43 expert dispatches. These are the target and draft vocabulary heads' `hcom_allGather` and `hcom_alltoallv`: gather hidden states, run the LM-head GEMM, then exchange logits.

The drafter blocks run once to produce base logits for the draft positions. The subsequent Markov head models token dependencies through sampling and embedding feedback. It does not rerun the transformer blocks for each proposal. The end of each step changes from MoE and attention to a serial chain of small projections and sampling operations.



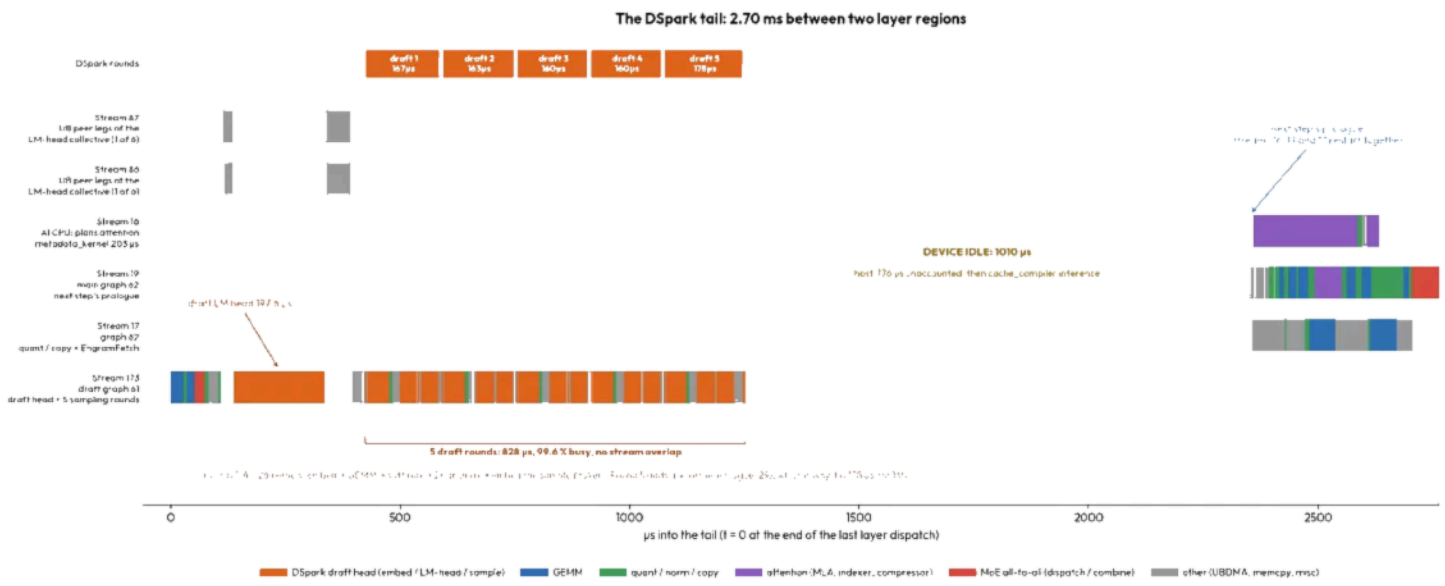


Source: SemiAnalysis



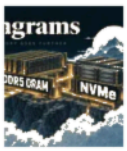
Source: SemiAnalysis

As mentioned and explained in previous articles, speculative decoding is a lossless method to predict draft tokens and use a single pass to verify them, amortizing weight loading. In the decode step diagram above, we can see how much faster DSpark token drafting is compared to the verification step.



Source: SemiAnalysis

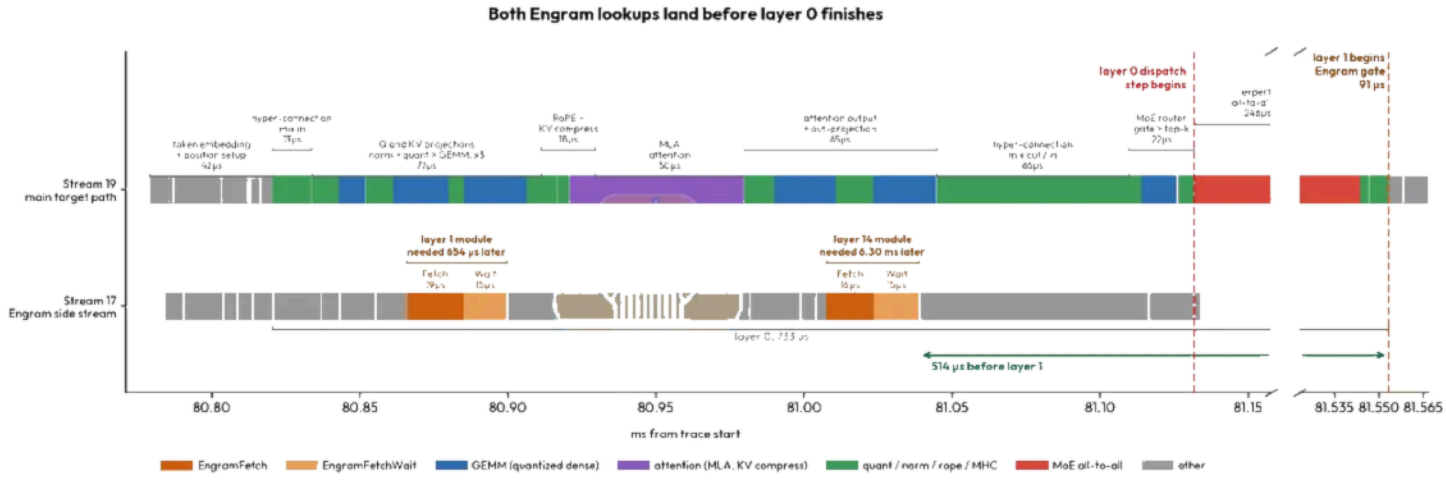
DSv4.1 Flash is also the first DeepSeek model to use Engram, but after Qwen and Meituan Longcat.



BRYAN SHAN, CAM QUILICI, AND 4 OTHERS · 9月18日

Read full story →

Because Engram takes token IDs as input, the fetch can be done async and overlapped with layer 0's compute. This allows offloading to DRAM without performance loss vis-a-vis HBM, but offloading to a PCIe 4.0 SSD degrades performance. Read more in our Engram analysis:

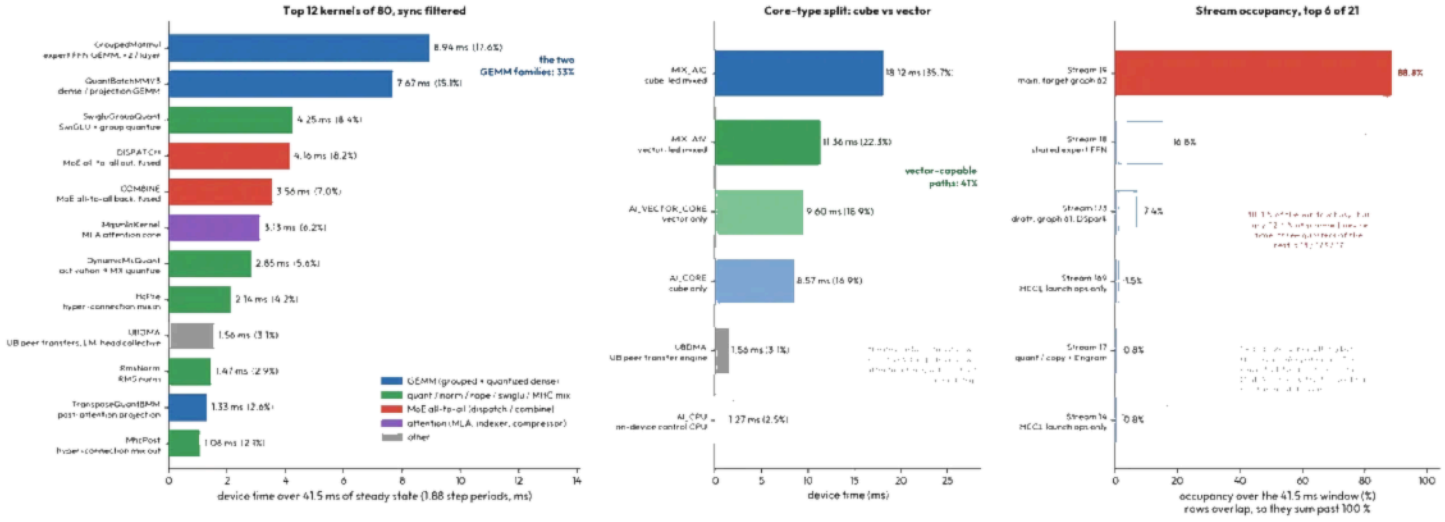


Source: SemiAnalysis

Summarizing kernel durations in the steady-state window, Stream 19 is occupied for 88.8% of the window but contributes 72.6% of summed device time because other

streams overlap it. In the middle, the substantial `MIX_AIV` and `AI_VECTOR_CORE` bands contain dispatch/combine as well as quantization.

Ascend exposes separately scheduled cube and vector paths through `MIX_AIC`, `MIX_AIV`, `AI_CORE`, and `AI_VECTOR_CORE`. Each event has a task type, so the duration accounting is a partition. Mixed types still involve work beyond a single physical unit; their names do not turn elapsed kernel duration into measured cube-only or vector-only utilization.



Source: SemiAnalysis

A first-order scenario doubling cube throughput while leaving other device work unchanged gives a 1.32× step improvement. It holds measured stream overlap constant and leaves idle unchanged. This is a composition scenario, not a prediction for new silicon: mixed task types are not exclusive utilization measurements, and bandwidth, dependencies, and scheduling can change the result.

Quantization fusion likewise remains an opportunity, its attraction is the repeated producer-to-consumer boundary and its large share of the vector bucket that also carries fused communication.

Ecosystem gaps

One issue however is the partnership could work extremely well for DeepSeek without making Ascend equally usable for everyone else.

For DeepSeek, building compilers, maintaining kernels and debugging vendor-specific behavior competes with model research and product development, albeit temporarily.

For everyone else, there may be issues when hardware configurations doesn't align with DeepSeek's. Most optimizations were on supernodes, and as of `vLLM-Ascend_0.26.0rc1` many issues exist working with Ascend on x86 hosts:

- No DeepGEMM support on x86 hosts
- Dependency on `memfabric_hybrid` containing hardcoded ARM headers for vLLM-Ascend, blocking startup; vLLM-Ascend doesn't work out of the box.
- CPU-binding skipped for non-ARM CPUs on vLLM-Ascend
- No NPU + x86 CI, patches needed for some recipes

Conclusion

Huawei already has highly efficient implementations of some foundational operations. DeepSeek brings model-specific workloads and engineering that extend the platform beyond those operations, while Huawei's architecture and native stack shape how those implementations are built. The resulting compiler and kernel work matters not only for today's performance, but for the engineering it can save on the next model and deployment.

A broader opportunity is a engineering multiplier for China's AI industry. Other operators serving DeepSeek can reuse kernels and deployment practices, while other labs can build new models on shared compiler capabilities. If these users contribute fixes and optimizations back, the gains can compound: each team starts with more of the difficult work already done.



Source:

On the Dwarkesh Podcast, Jensen talked about his concerns with chip export restrictions. From the software results so far and testing on 950PR, the ecosystem gap is definitely closer with DeepSeek's help on top of Huawei's work, Jensen's nightmare is starting to come true.

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



147 Likes · 5 Restacks

Restacks

Write a comment...

© 2026 Dylan Patel · Privacy · Terms · Collection notice
Substack is the home for great culture