The Huawei-DeepSeek Symbiosis

华为与 DeepSeek 的共生关系

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

华为昇腾 950PR 深度解析与微基准测试、DSv4.1 快速分析、DeepSeek 对昇腾的支持、AICPU、Tilelang 昇腾版

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

华为昇腾 950PR 深度解析与微基准测试、DSv4.1 快速分析、DeepSeek 对昇腾的支持

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

就在中国国庆假期前夕，DeepSeek 发布了对昇腾的支持，涵盖 TileLang、DeepGEMM、DeepEP、TileKernels、FlashMLA 以及 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.

双方单凭自身都无法做到这一点。尽管拥有盘古模型，但华为缺乏能够证明其芯片能够胜任前沿模型训练或推理的顶级实验室；DeepSeek 拥有前沿模型，却无法直接采购所需算力。双方恰好填补了彼此的空白：华为提供硬件平台与原生软件栈；DeepSeek 则带来了使其硬件能够有效服务于模型所需的真实工作负载与工程能力。9

月 30 日 DeepSeek 在其代码仓库中发布对昇腾的支持，正是这场深度协同的产物，而这种协作正是通过算子编写语言串联起来的。



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.

到 2028 年甚至 2027 年，DeepSeek 很可能将主要使用华为的技术栈，尤其是在内外部压力都推动构建全自主中国生态的背景下。正如我们在这里以及后续文章中所展示的，华为生态已经成熟且发展完善，从昇腾（Ascend）时代起就打下了坚实的基础，DeepSeek 的工程师们正在此之上继续深耕。

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.

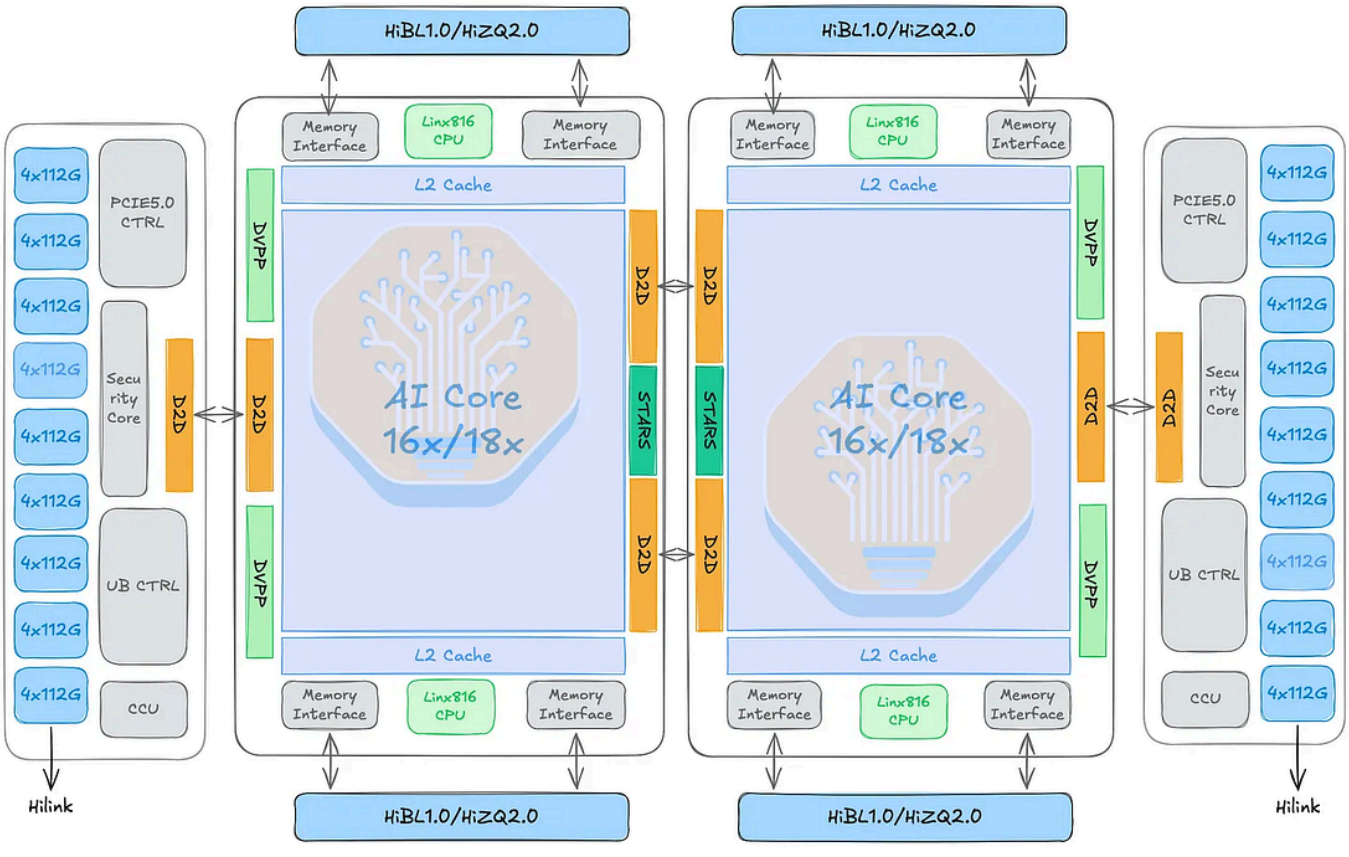
在本文中，我们将首先剖析昇腾 950PR 的架构，并对其算力和内存性能进行基准测试。随后，我们将探讨 DeepSeek 的软件发布成果，从 TileLang 的编译器能力，到针对特定硬件的算子优化，再到机柜级通信。最后，我们将分析 DeepSeek-V4.1-Flash 的执行特征。

Ascend 950

昇腾 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.

我们在《DeepSeek V4 Pro》一文中已深入剖析过 CANN 和昇腾 950 的架构。自那以后，又有更多值得关注的细节被披露出来。



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.

从芯片本身来看，950PR/DT 已经出现了多种核心数量、HBM 堆叠层数及缓存大小各不相同的衍生版本。其中部分版本可能仅为虚拟化/模拟规格，或仅供测试使用。这是因为，受完全物理缺陷以及功耗和性能层面的参数未达标影响，仅有一小部分裸片和 HBM 堆叠能够达到目标规格；因此，华为对算力裸片和内存同时进行分档筛选（binning），划分为多个 SKU，并将体质较弱的内存与性能较弱的算力单元搭配使用，而不是像英伟达那样仅对算力裸片进行分级筛选。

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

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.

这并非新鲜事，910B 也有多个变体，例如 910B1、910B2、910B3、910B4。在下面的微基准测试中，我们使用的是一块 Ascend 950PR_9579，配备 28 个 Cube 核心和 128 GiB HBM。

```
root@cb3bb851f93f:/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(page) |
| | | 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 |
+===========================+==============+==============================
======================+
```

root@cb3bb851f93f:/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(page) |

| | | Bus-Id | NPU Util(%) Memory-Usage(MB) HBM-Usage(MB) |

+========+================+==============+===============================

==================================+

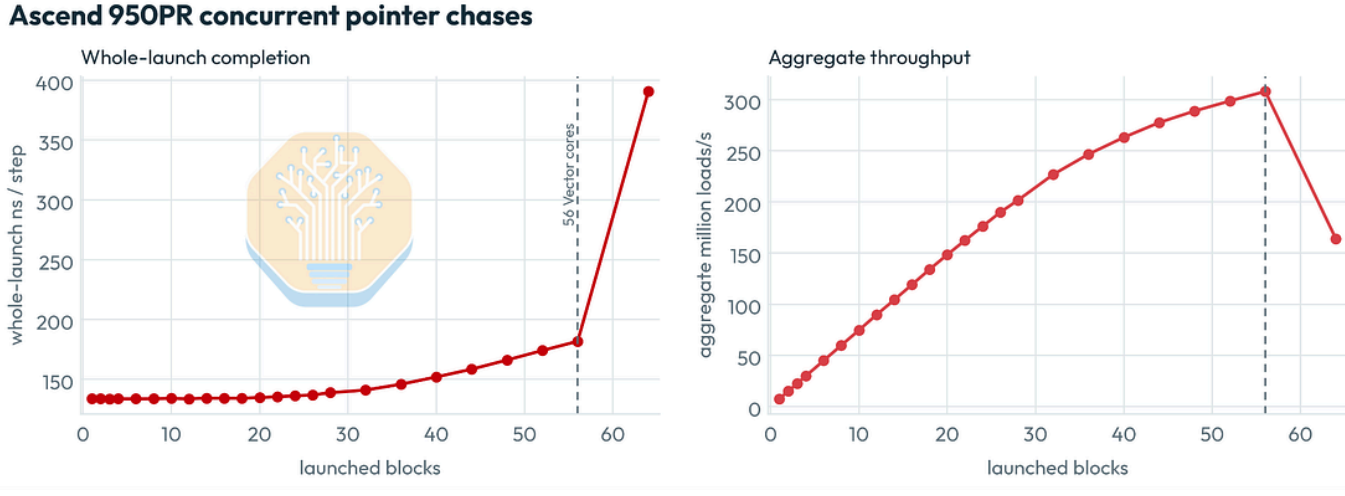
| 0 | Ascend950PR | OK | 203.3 56 0 / 0 |

| | | 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 |
+============================+==============+=================
==================================+
```

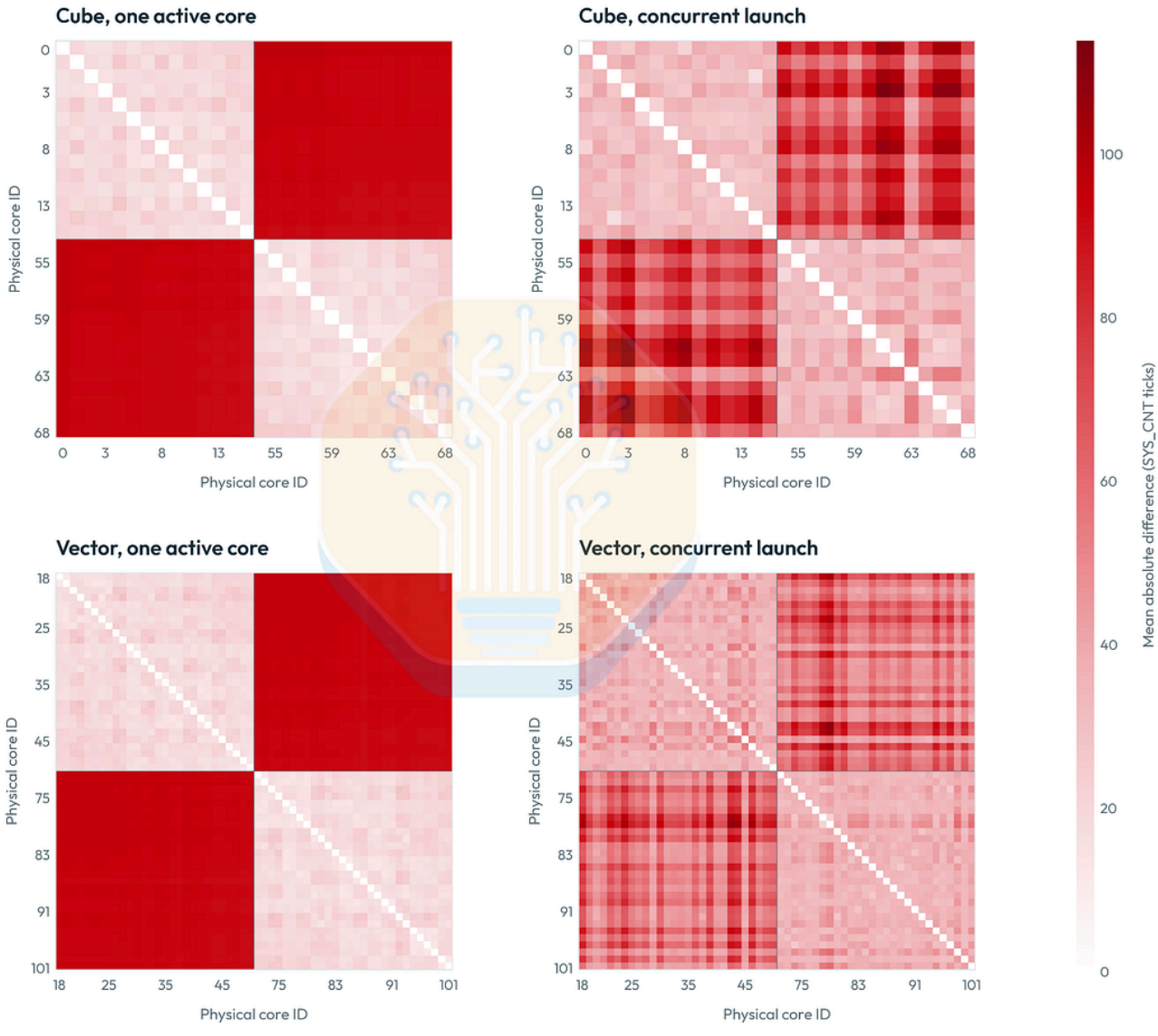
Running point chases reading 16 KiB of integer payload per launch, it stays near 134ns through 18 blocks, rises to 182ns at 56, then jumps to 391ns at 64. This is consistent with a second wave beyond 56 Vector cores.

点追逐（point chase）测试每次启动读取 16 KiB 的整数数据，在 18 个 block 以内延迟保持在 134ns 左右，到 56 个 block 时升至 182ns，到 64 个 block 时则跳升至 391ns。这与超过 56 个 Vector 核心后启用第二波核心的情况相符。



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.

随后，通过在每个 Cube 和 Vector 核心上分别运行标量指针追踪探针，记录贯穿整条 128 MiB 链上匹配地址的加载耗时，我们得到了下图。每个单元格的颜色代表两个核心地址访问特征之间的平均绝对差，由此呈现出与解析出的裸片相吻合的两个组。



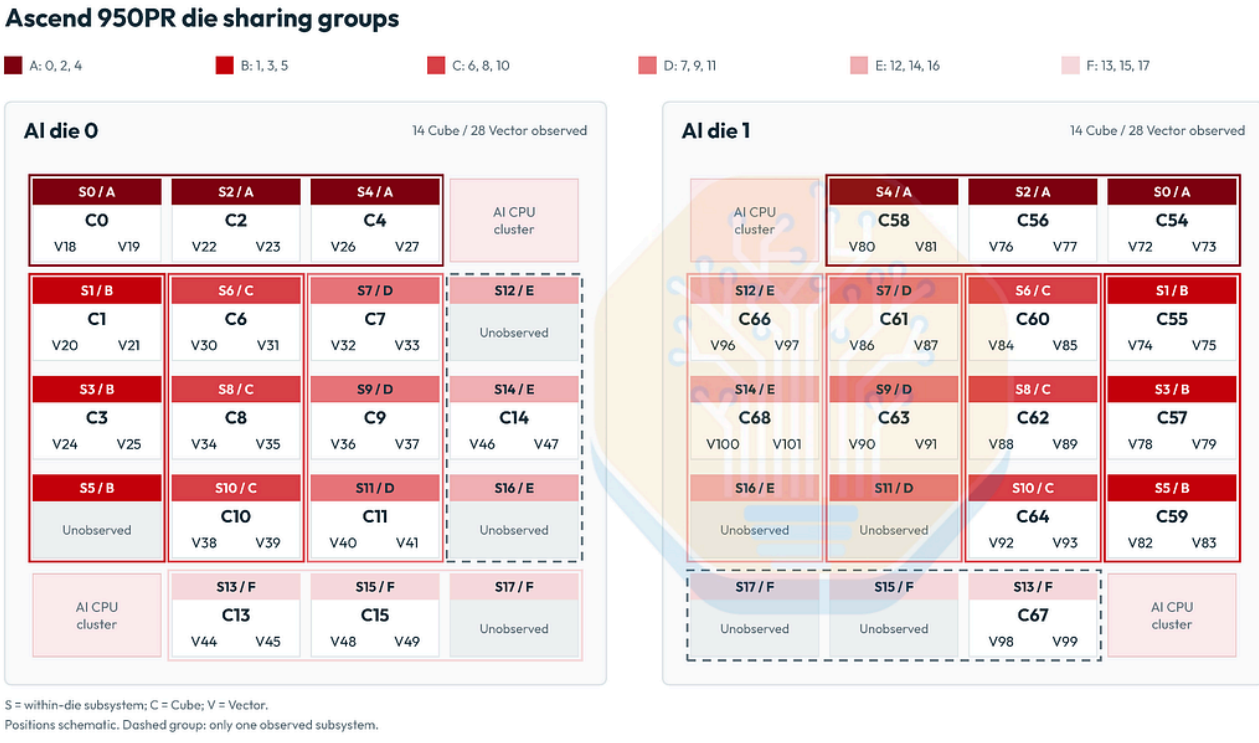
Source: SemiAnalysis

来源：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.

进一步来看，通过在另一个核心运行内存访问流量或纯寄存器负载的同时，对某个 Vector 核心的指针追逐（pointer chase）进行测时，所产生的额外降速揭示了哪些子系统存在资源共享，从而印证了诸如 {0,2,4} 和 {1,3,5} 这样的分组。Cube 与 Vector

的混合启动则识别出了每个子系统各自包含一个 Cube 和两个 Vector；我们在这些分组中将它们绘制在一起，物理布局在此仅作示意。



Source: SemiAnalysis

来源：SemiAnalysis

CPU within the NPU

NPU 内部的 CPU

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:

每颗 950PR 均配备可变数量的 Linx816 AArch64 ELF AICPU。MIDR 编号表明，这些核心很可能与麒麟 9000S/9010 中的中核相同。这些 AICPU 核心具备以下指令集支持：

```
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 ssbs sb paca pacg dcpodp flagm2
frint svei8mm svebf16 i8mm bf16 dgh rng ecv
```

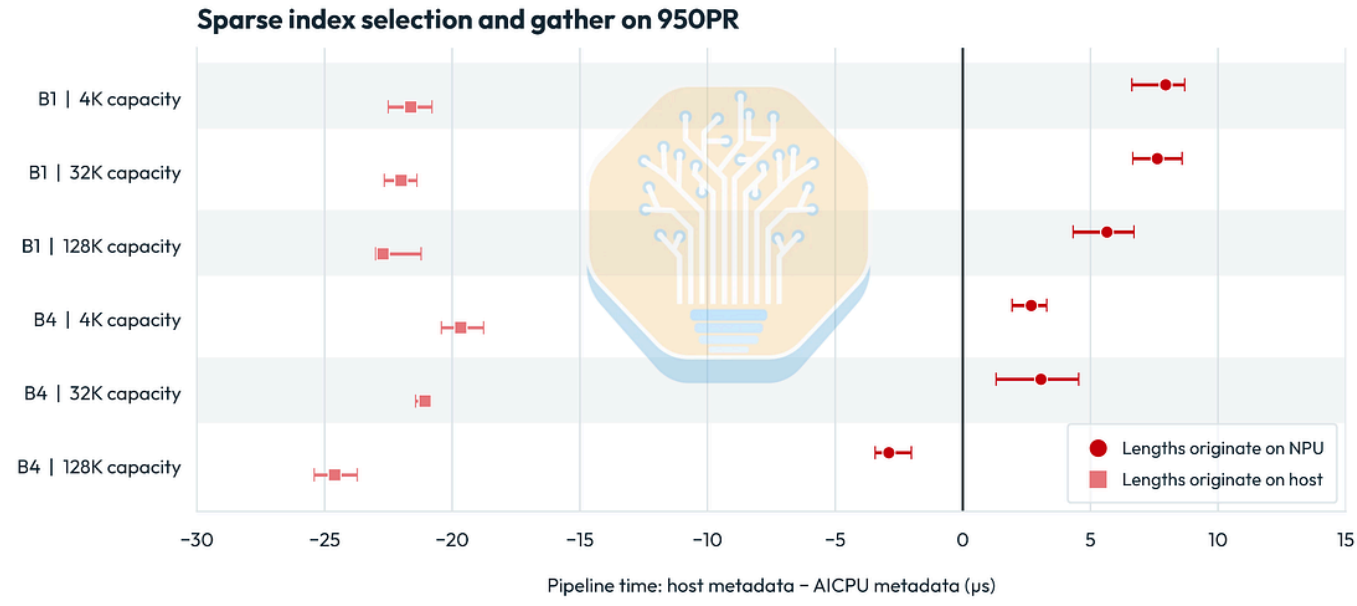
fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp cpuid
asimdrdm jscvt fcma lrcpc dcpop sha3 sm3 sm4 asimddp sha512 sve
asimdfhm dit uscat ilrcpc flagm ssbs sb paca pacg dcpodp flagm2
frint svei8mm svebf16 i8mm bf16 dgh rng ecv

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.

针对该 SKU，共配置了 7 个 AICPU 核心，其中 6 个可见，5 个参与实际执行。除设备控制外，它还能提供 CPU 算子支持，例如处理 KV 元数据。对于 D2H（设备到主机）传输开销占据绝大部分耗时的短任务，AICPU 能够有效发挥辅助作用。

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.

在 DSA 中，一个微小的调度步骤会读取当前序列长度，并将索引器（indexer）任务分配给加速器核心。随后 Cube/Vector 核心计算 INT8 索引器得分，并筛选出前 2,048 个索引。在 vLLM-ascend 的实现中，这些规划、索引选择和 gather 步骤均在 AICPU 上运行，我们将其性能与在主机 CPU 上运行的情况进行了对比。



Source: SemiAnalysis

来源：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.

由于算力性能有限，AICPU 只适合开销占主导的短任务，以及数据在 NPU 上产生并在 NPU 上被消费的工作负载。

Microbenchmarks

微基准测试

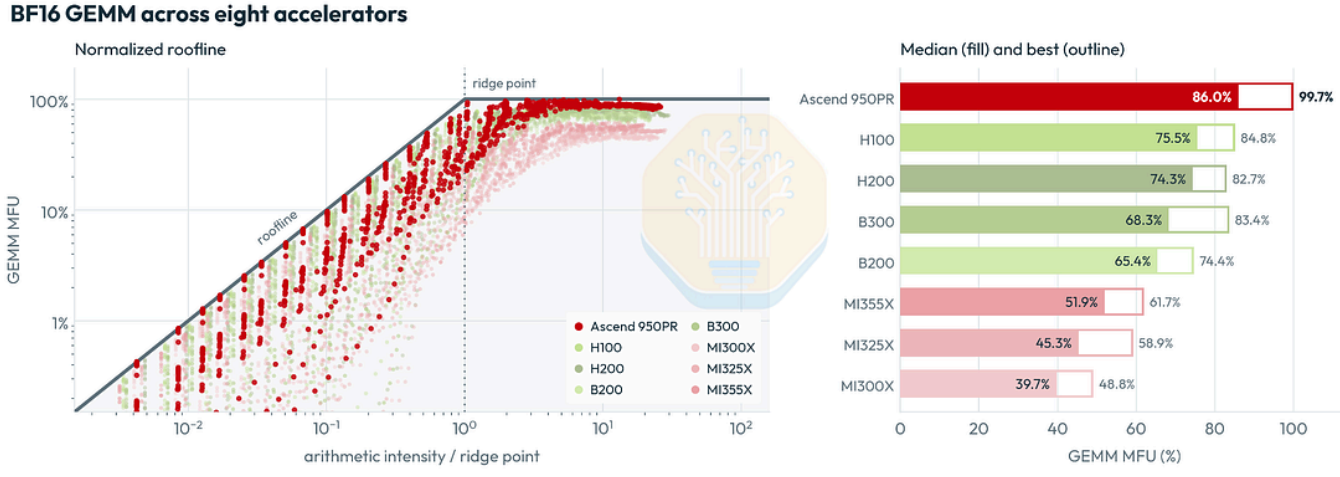
Ascend 950PR is designed for prefill and recommendation. It gets lower memory bandwidth and core than its bigger brother the 950DT.

Ascend 950PR 专为预填充（prefill）和推荐场景设计。相比更大的兄弟型号 950DT，它的内存带宽和核心规格都更低。

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.

下面的结果并非来自 DeepSeek 的贡献，而是来自昇腾上原生的 `torch.matmul`，其解析到底层为 `aclnnMatmul_MatMulV3Common_MatMulV3`。我们对比了来自多个开源模型的 1,055 个 BF16 GEMM 形状。华为最优的算子达到了理论上限的 99.7%，这表明 DeepSeek 通过 DeepGEMM 实现的 99%+ 利用率并非最大化利用率的唯一途径。华为工程师同样做了出色的工作。



Source: SemiAnalysis

来源：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.

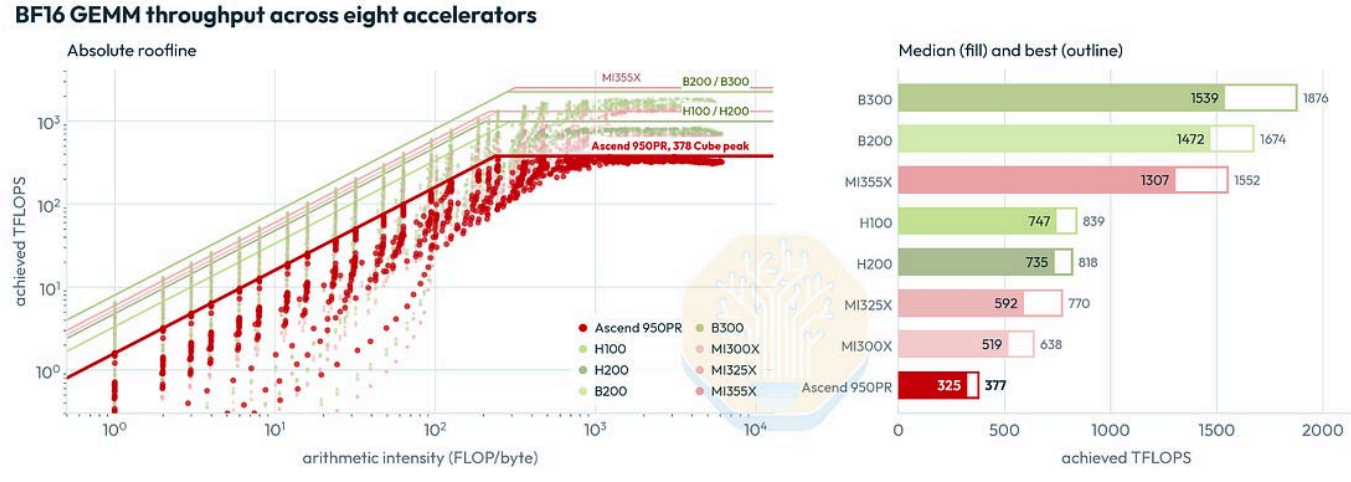
这种优势主要集中在计算密集型任务上。在处于昇腾算力拐点（ridge）之下的矩阵形状测试中，其达成的实际/理论可达利用率比例中位数为 41.5%，而 H100 为 46.8%。英伟达的部分测试结果受到了功耗限制，未达到最高 SM 频率：在 700 W 的功耗上限下，其中位数功耗约为 690-698 W，采样的 SM 频率中位数在 1335-1470 MHz，低于 1830 MHz 的最高值。华为的功耗估算约为 599-600 W，运行频率为 1500-1550 MHz，低于其 1650 MHz 的基准参考值。

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.

昇腾的 MXU 设计可能是实现这种高利用率的原因之一。昇腾将矩阵操作数暂存在专用的 L0A/L0B 缓冲区中，并将累加值保存在 L0C 中，使数据搬移、计算与结果处理得以重叠并行。华为认为，增大的 L0C 能够带来更灵活的分块（tiling）与数据复用。

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

但毋庸置疑，950PR 在绝对性能数值上依然落后，处于 A100 与 MI300X 之间的水平。



Source: SemiAnalysis

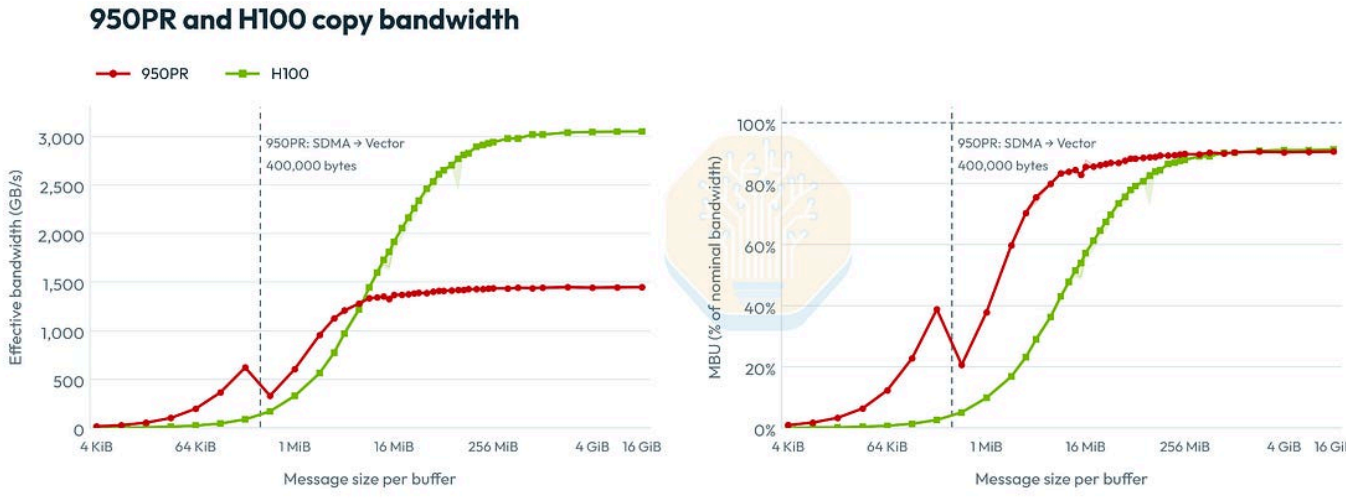
来源：SemiAnalysis

MemBW

内存带宽（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.

在测试 HBM 内存带宽时，我们发现，在每个缓冲区的消息尺寸较小时，950PR 的有效带宽和带宽利用率均超过了 H100；而在消息尺寸较大时，其带宽利用率也几乎与 H100 持平。



Source: SemiAnalysis

来源：SemiAnalysis

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.

曲线中出现了一处凹陷，这是由于昇腾切换了复制方式，转为调用启动开销更大的 Vector 核 TensorMove 算子，导致数据量稍大一些的复制操作耗时反而突然增加。而对于超大规模的数据复制，这部分额外开销占比相对变小，因此带宽又会重新回升。

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.

需要注意的是，高 FLOPs 利用率和高带宽利用率并不一定会转化为高 MFU/MBU。模型训练（尤其是采用 FSDP 时）需要进行集合通信、检查点保存以及数据搬运，而这些操作往往很难被完全重叠掩盖。

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.

正如我们在关于 Jalepeno 的文章中提到的，基准测试里的“芯片”概念不过是一种人为定义。多个裸片（die）完全可以通过封装拼接成一颗更大的“芯片”。因此，每瓦性能或总拥有成本（TCO）才是最合理的对比基准。但对中国来说，电力成本要便宜得多。我们将在后续文章中进一步分析昇腾单位 TCO 的性能表现。



OpenAI Jalapeño：表现优于英伟达 Blackwell

OpenAI Jalapeño: Better Than Nvidia Blackwell

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.

硬件提高了上限，但软件决定了你能否触达那个上限。尽管华为的硬件可能落后于 A100 超过约 5 年，但软件已经处于前沿水平。

DeepSeek Open Source

DeepSeek 开源

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.

在获得昇腾支持的 6 个代码库（TileLang、DeepGEMM、DeepEP、TileKernels、FlashMLA、DeepSelect）中，截至撰稿时只有 TileLang 可用，且 CANN 9.2.0 尚未发布。DeepSeek 要求使用 CANN 9.2.0，该版本在本文撰写之后才发布，并且需要 C++20 编译器，而 CANN 9.2.0-beta.2 并未提供。HDK 将于 10 月 15 日发布。这发生在 DeepSeek 2025 年开源周之后，当时这些代码库中有部分发布时仅支持 Nvidia。

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.

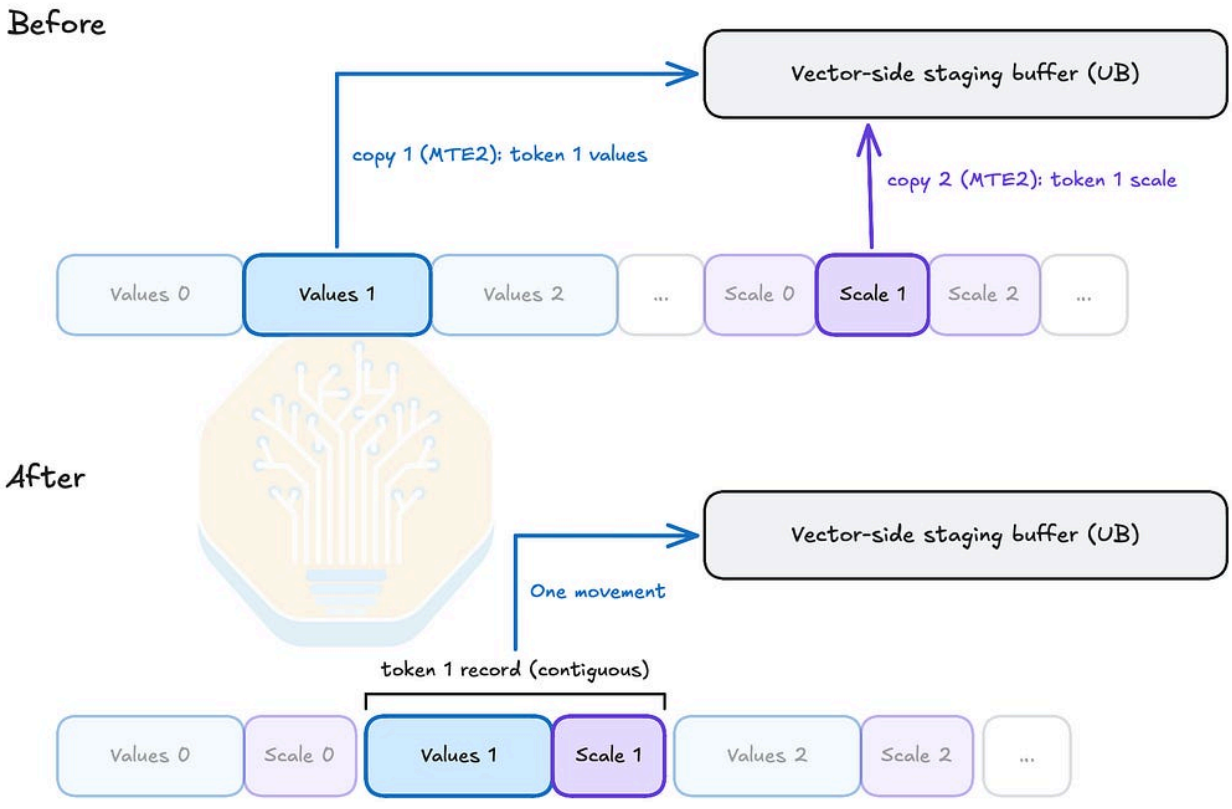
从上述结果可以看出，华为已经能够良好地执行相关运算；这种共生关系的意义在于，让应用的其余部分以及支持它所需的工作变得更高效、更可复用。

Engineering runs both ways

工程协作是双向的

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).

华为的架构已经开始影响 DeepSeek 软件组织数据的方式。早期 FlashMLA 的 KV 缓存布局将量化值与其缩放因子存放在不同区域。在昇腾上，取回一个 Token 就需要为其数值和缩放因子分别进行数据传输。新实现将两者放在一起，从而减少了内存传输引擎（MTE）的传输请求次数。



Source: SemiAnalysis

来源：SemiAnalysis

From fast kernels to reusable kernel engineering

从高效算子到可复用的算子工程

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.

Ascend 950 后端已合并至 TileLang 主代码库中，而非仅以独立的华为适配器形式存在。它针对 Ascend 的执行模型，补充了原生编译、调度、同步以及运行时集成支持。

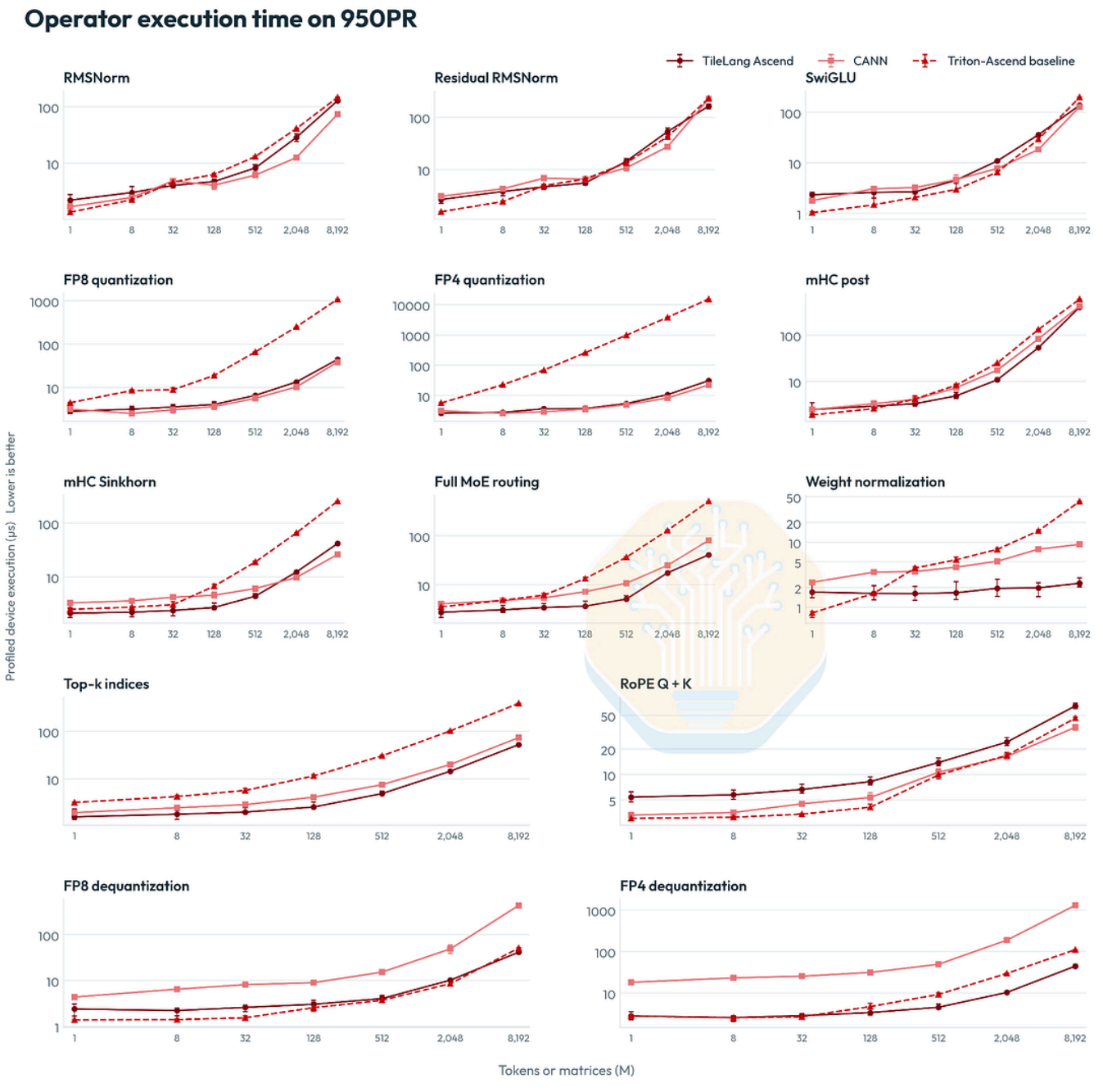
TileLang makes Ascend kernels easier to write, but developers still need to account for the hardware. They choose the calculations, tile sizes and where data is stored.

The compiler handles much of the coordination between Cube cores, Vector cores and data transfers.

TileLang 让 Ascend 算子的编写变得更为简便，但开发者仍需充分考量底层硬件特性。开发者需要决定计算逻辑、分块大小（tile size）以及数据存放位置；而编译器则负责处理 Cube 核心、Vector 核心与数据传输之间的大部分协同工作。

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.

进一步深入底层实现，并对比其与 CANN 及 Triton-ascend 的性能表现可以看到，TileLang 在某些场景下能够匹敌甚至超越 CANN；最优实现方案会随着计算规模的变化而变化。需要说明的是，这只是一个在缓存预热状态下进行的简易基准测试。

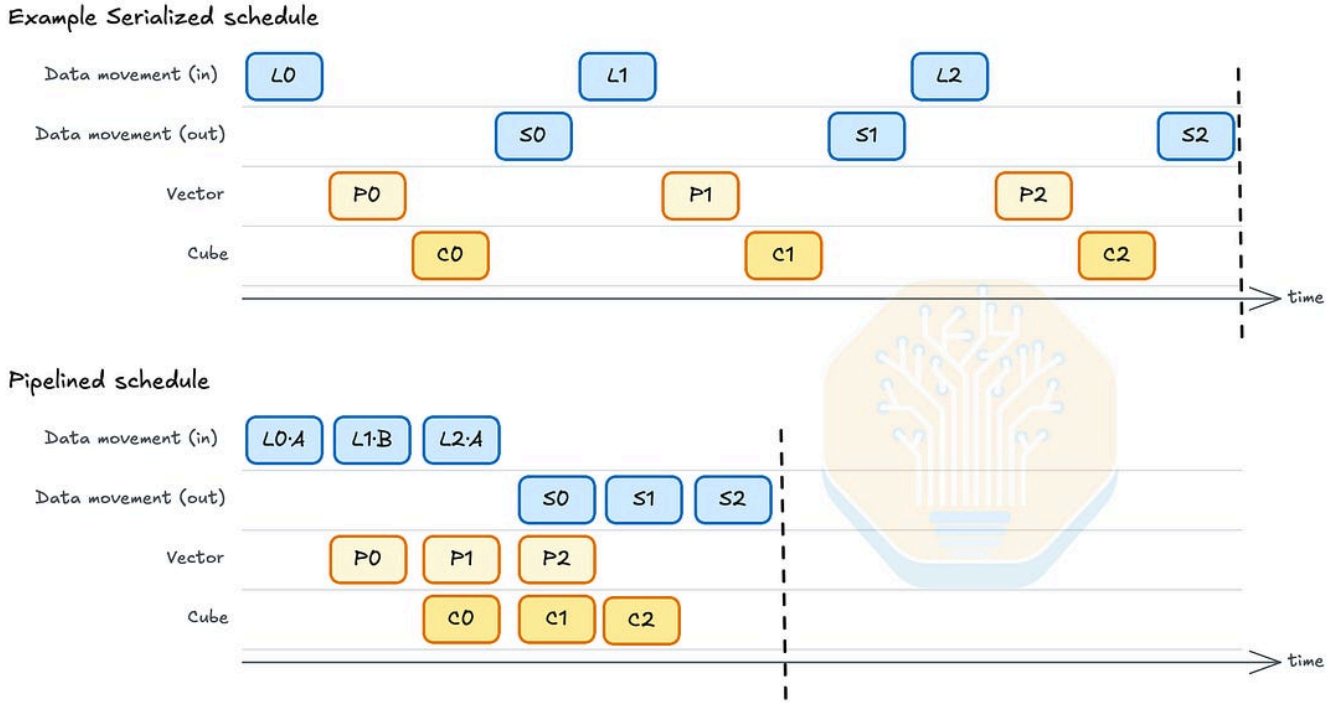


Source: SemiAnalysis

来源：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.

AutoSchedule 利用数据依赖关系与预估执行时间，规划哪些操作可以重叠执行。它还会在可用的片上内存中决定使用多少个缓冲槽，从而在处理当前分块（tile）的同时预先加载后续数据。当默认调度不适用时，开发者也可以对这些决策进行干预调整。

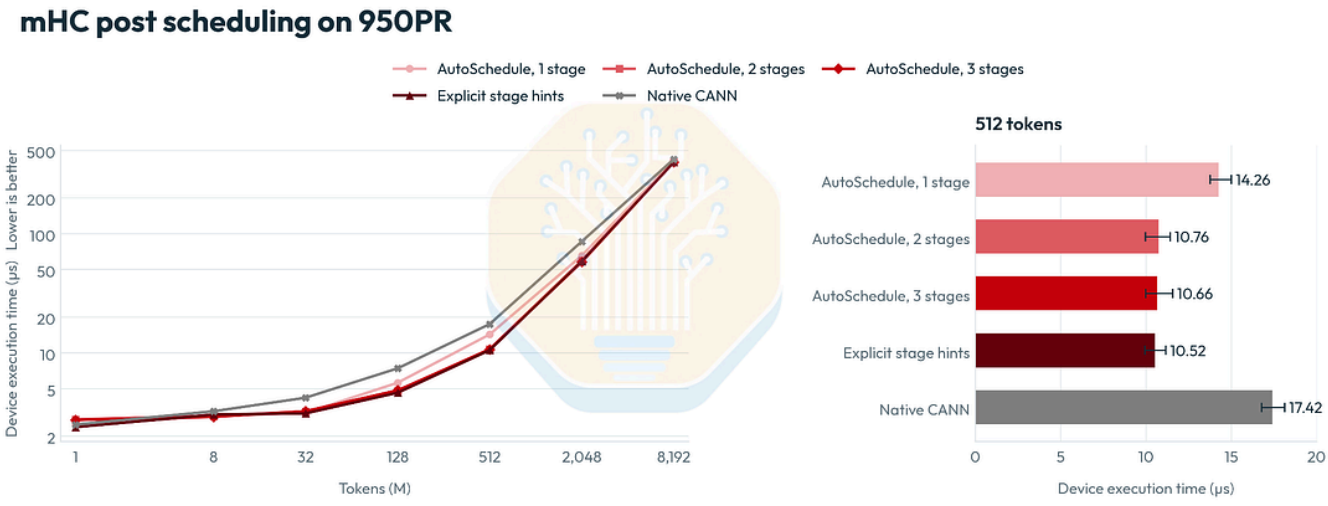


Source: SemiAnalysis

来源：SemiAnalysis

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

在 950PR 上，AutoSchedule 在较小规模下效果尤为明显，可将单项操作的执行时间缩短多达 40%。



Source: SemiAnalysis

来源：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.

编译器还会插入同步机制，确保各操作等待其所需的数据，同时消除那些因其他依赖关系而变得不必要的等待。它还会检查条件分支，避免在某个分支可能根本不会执行的情况下预设操作已完成。这减轻了程序员必须自行管理的同步负担。

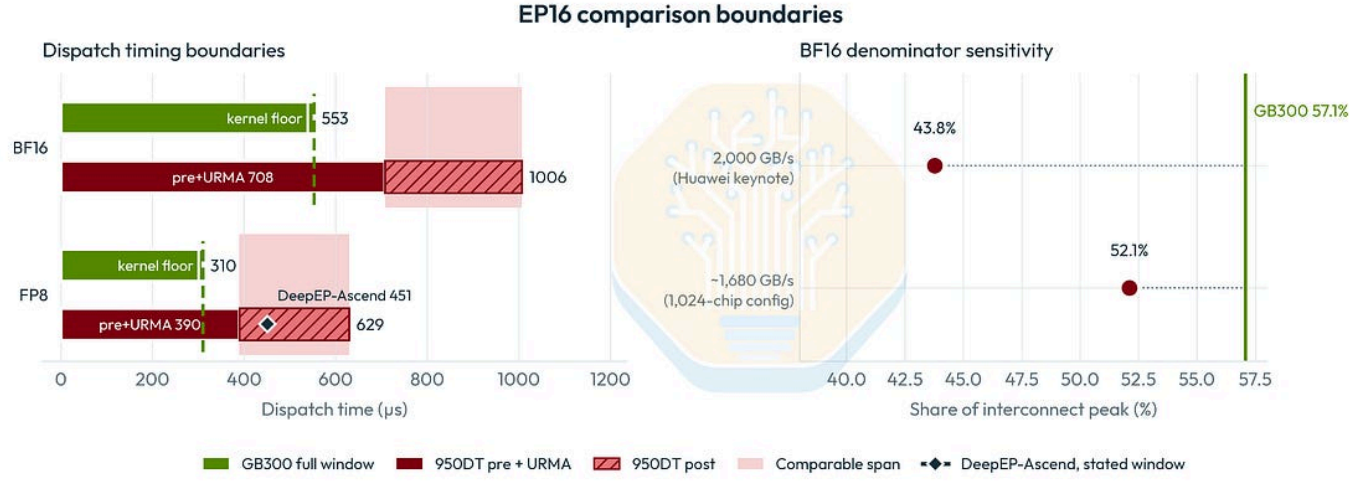
DeepEP

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.

华为已与英伟达一同将大型纵向扩展（scale-up）AI 系统推向商业化，其昇腾（Ascend）超节点将紧密耦合的加速器互连跨多机柜延伸，从 910 CloudMatrix384 拓展到 950 SuperPod，从而实现了大范围专家并行（wide-EP）的横向扩展。9 月



30 日的发布中，华为还公布了在 950DT 超节点上通过 URMA 运行的 EP16 dispatch/combine 基准测试结果。



Source: Huawei, DeepSeek, SemiAnalysis

来源：Huawei、DeepSeek、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.

华为采用三个串行的算子（前处理/URMA/后处理），而 GB300 是一次融合调用；华为的后处理包含接收完成（receive completion）环节——我们的测量已将其计入——同时还包含按专家进行的扩展操作，而 GB300 的结果并未包含这一部分。

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.

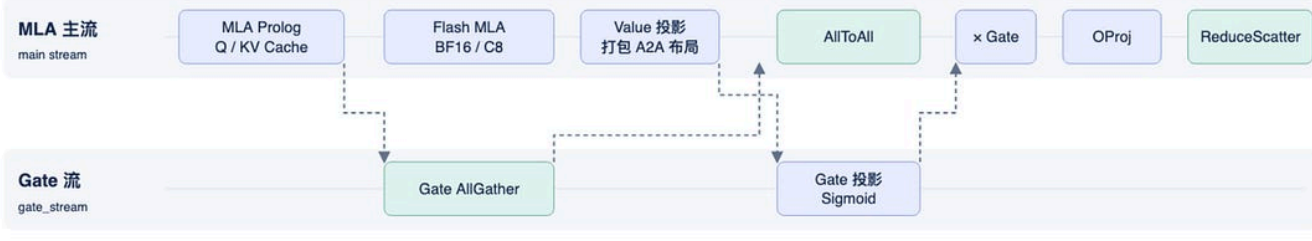
DeepEP-Ascend 是 DeepEP V2.5 的移植版本，保留了相同的 Python API，但底层改用 URMA 和 HCCL。2025 年 7 月，华为昇腾团队曾提议将 DeepEP-Ascend 纳入 `deepseek-ai`，但被 DeepEP 的维护者拒绝。如今，DeepEP-Ascend 已改为在 DeepSeek 自己的组织下发布。

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.

昇腾将 Cube 与 Vector 执行路径分离，这为通信与计算的重叠执行创造了机会，但这种重叠并非没有代价。专家计算同样可能需要向量处理和数据搬运，而通信会争用共享内存带宽。因此，在不拖慢并行运行的计算的前提下，能够隐藏多少通信，始终存在一个权衡。

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: Huawei

来源：华为

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.

这种资源平衡问题并非 DeepSeek 所独有。华为的 Kimi K3 方案明确限制了向量单元的使用，以便为并发任务预留算力，并根据需要重叠执行的任务，在专用 CCU 与基于

向量单元的通信之间进行权衡取舍。仅仅提升单个操作的执行速度是不够的：各项操作对资源的需求也必须彼此契合。

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.

DeepEP-Ascend 的文档指出了组合处理（combine processing）与 URMA 通信在争夺 HBM 资源时的冲突，并表示华为计划通过固件改进来缓解这一问题。这就是软硬件协同设计（co-design）的实际体现：模型暴露瓶颈，工程实现针对性调整，而平台的开发优先级则跟进解决遗留问题。

All planned

尽在计划之中

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

在 5 月份一份未经验证的 DeepSeek 投资者会议纪要中，梁文锋表示：

> "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."

> "因此，谈到中美之间的芯片差距，我认为软件生态上的差距终将消除。但在硬件层面，我们面临两个不同的劣势：需要四张华为加速卡才能匹敌一张英伟达加速卡，同时华为在硬件推向市场的进度上也落后大约两年。"

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

4 个月后，生态差距确实有所缩小。同一份纪要中还提到了 TileLang，以及协助华为构建完善生态的相关内容。

DSv4.1 Flash Optimizations

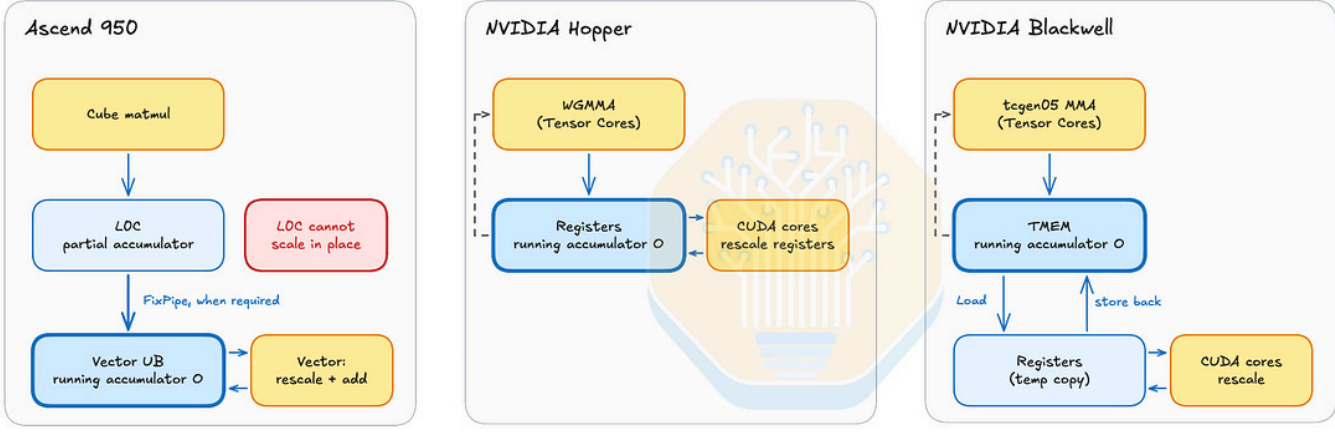
DSv4.1 Flash 优化

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.

DeepSeek 与华为共生关系的另一个体现，在于 DeepSeek 会向华为提供模型权重，从而支持首日即优化的版本发布，例如 DeepSeek V4 和 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

昇腾（Ascend）的累加器设计迫使其在注意力计算上采用不同的处理方式。Cube 的 L0C 缓冲区无法就地对累加输出进行重新缩放，而将重新缩放后的输出传回 L0C 又需要执行开销高昂的单位矩阵乘法。DeepSeek 则转而将运行中的输出累加器保留在 Vector UB 中，仅在需要重新缩放和累加时才从 L0C 转移部分和（partial sums）。相比之下，Hopper 将其累加器保存在寄存器中，而 Blackwell 则支持张量内存与寄存器之间的加载和存储，从而规避了这一特定限制。



Source: SemiAnalysis

来源：SemiAnalysis

Profile

性能剖析

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.

华为还公布了在 32 颗芯片的华为昇腾 950DT（Huawei Ascend 950DT）部署环境下，DeepSeek-V4.1-Flash 单个 rank 及其主机线程的性能剖析（profile）数据。今年 6 月，我们曾分析过一份类似的性能剖析数据，即 DeepSeek V4 Pro 的相关情况。



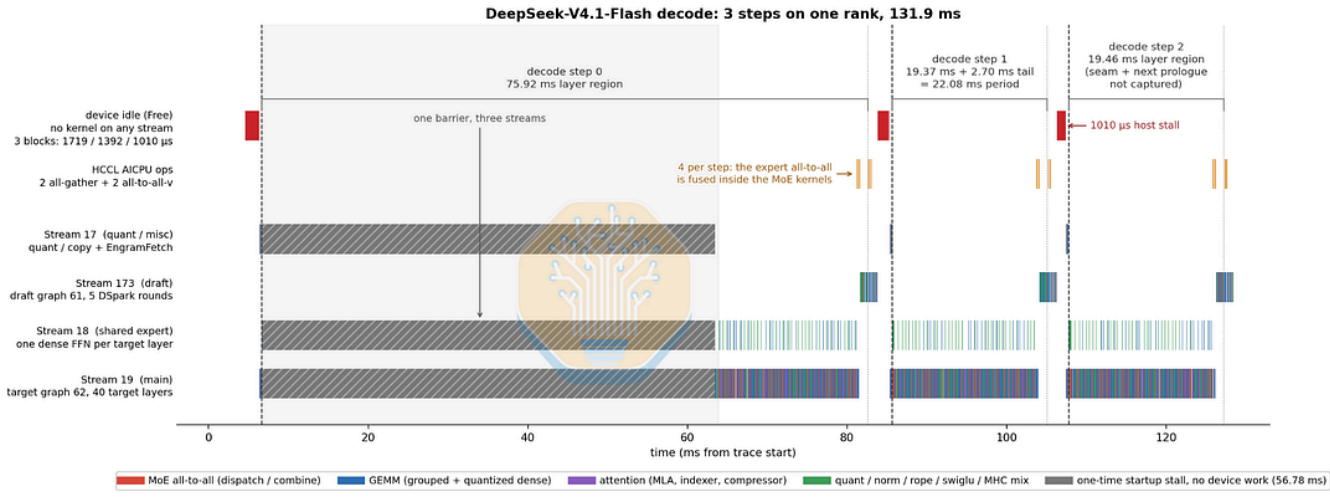
DeepSeekV4 1.6T 第 0 天至第 43 天性能随时间演进——华为、GB300 NVL72、MI355X、B200

DeepSeekV4 1.6T Day 0 to Day 43 Performance Over Time - Huawei, GB300 NVL72, MI355X, B200

Read full story

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

许多组件是相似的，例如 AICPU 的使用，以及将部分层分配到不同流（stream）上的做法。



Source: SemiAnalysis

来源：SemiAnalysis

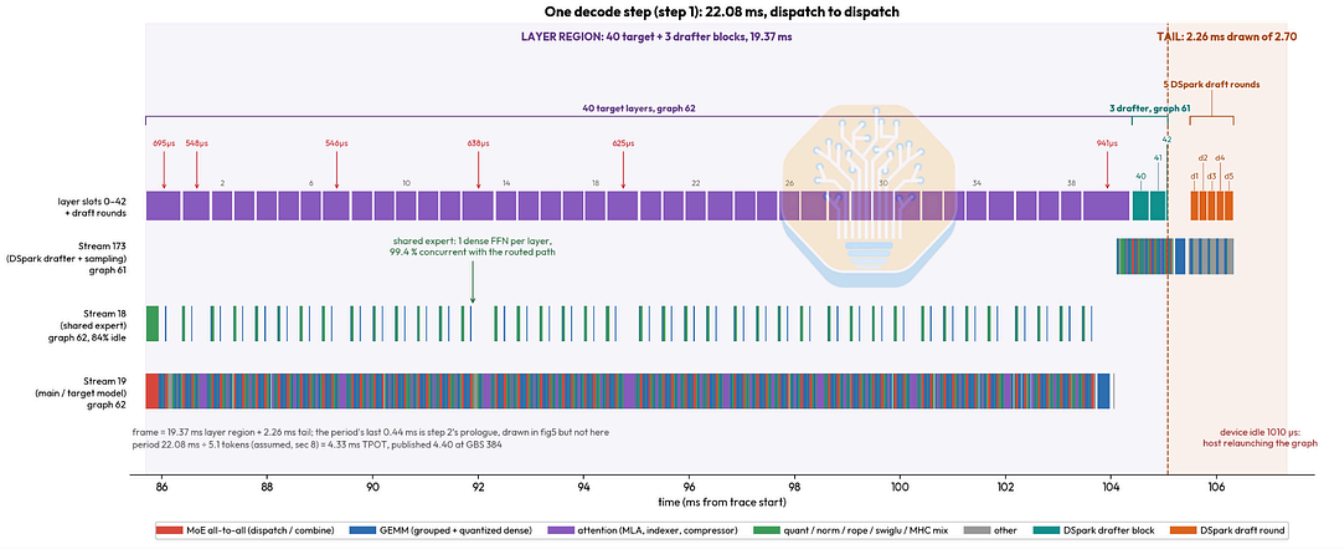
The execution order is 40 target-model layers, the target LM head, 3 DSpark drafter blocks, the draft LM head, and 5 sampling rounds. The main compute runs on Stream 19; the drafter's run on Stream 173. The V2 MC² kernels that fused computation and communication from the DSv4 Pro trace shows up again, 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.

执行顺序依次为：40 个目标模型层、目标模型 LM Head、3 个 DSpark 起草模块、起草模型 LM Head，以及 5 轮采样。主计算运行在 Stream 19 上；起草模型的计算则运行在 Stream 173 上。此前在 DSv4 Pro 的 Trace 中见过的计算与通信融合 V2 MC² 算子，此次以 V3 版本再次出现。这也解释了为何在 43 次专家分发中，每个步骤仅出现 4 次由 AI CPU 调起的 HCCL 操作。这些操作分别对应目标词表头与起草词表头的 `hcom_allGather` 与 `hcom_alltoallv`：聚合隐藏状态、执行 LM Head 的 GEMM 计算，随后交换 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

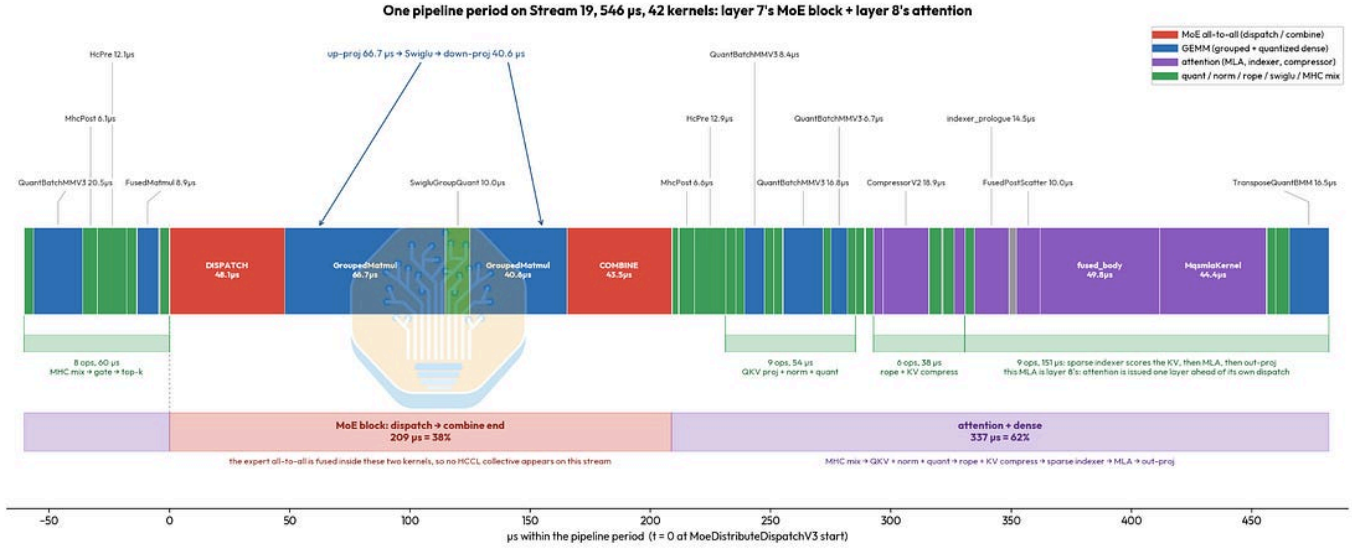
end of each step changes from MoE and attention to a serial chain of small projections and sampling operations.

起草模块仅运行一次，用于为起草位置生成基础 logits。随后的马尔可夫头（Markov head）则通过采样与嵌入反馈来建模 Token 间的依赖关系，而无需为每个候选 Token 重新运行 Transformer 模块。每个步骤尾部的计算形态因此从 MoE 与注意力机制，转变为由一系列小型投影与采样操作构成的串行链。



Source: SemiAnalysis

来源：SemiAnalysis

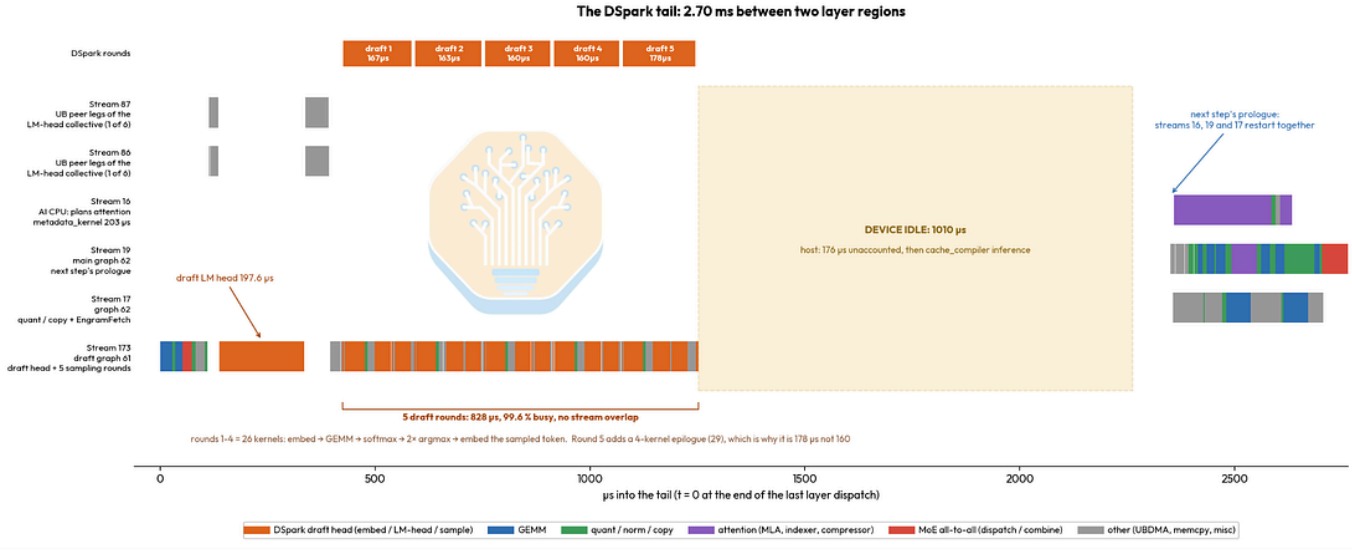


Source: SemiAnalysis

来源：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.

正如我们在之前的文章中所提及并解释的那样，投机性解码（speculative decoding）是一种无损方法，它通过预测候选 Token 并通过单次传递完成验证，从而分摊权重加载的开销。从上方的解码步骤示意图中可以看出，DSpark 的 Token 候选生成速度比验证步骤要快得多。



Source: SemiAnalysis

来源：SemiAnalysis

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

DSv4.1 Flash 也是首个采用 Engram 的 DeepSeek 模型，不过在此之前通义千问（Qwen）和美团 Longcat已经使用了这一技术。



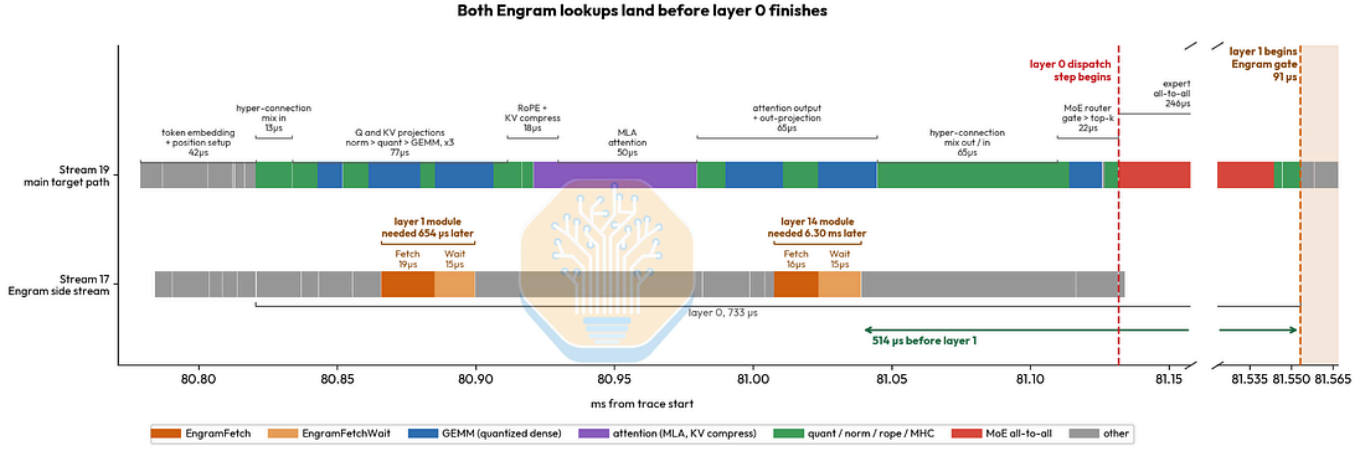
Engrams 嵌入的双重意蕴：面向高效 DRAM/SSD 卸载的协同设计

Engrams Embedding Entendre: Codesign for Efficient DRAM/SSD Offloading

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:

由于 Engram 以 Token ID 作为输入，因此取数可以异步进行，并与第 0 层的计算相互重叠。这使得将其卸载到 DRAM 时，相较于 HBM 不会出现性能损失；但若卸载到 PCIe 4.0 SSD，则会导致性能下降。详见我们的 Engram 分析：



Source: SemiAnalysis

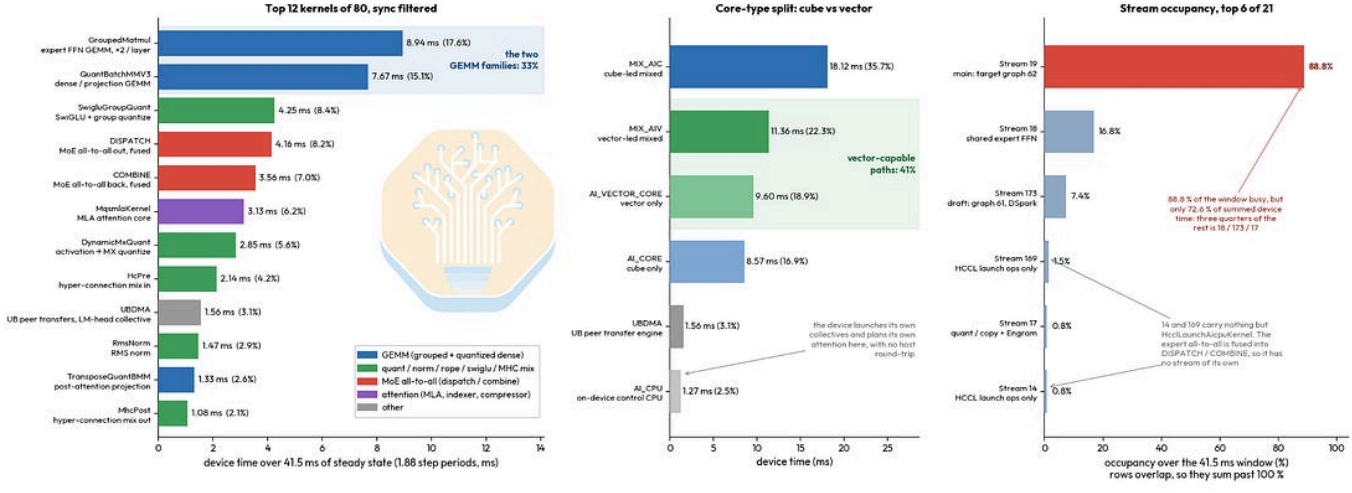
来源：SemiAnalysis

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.

对稳态窗口内的 kernel 时长进行汇总：Stream 19 在该窗口中的占用率为 88.8%，但由于其他 stream 与其重叠，其贡献的设备时间总和占比为 72.6%。图中中部大片显著的 `MIX_AIV` 和 `AI_VECTOR_CORE` 区带包含 dispatch/combine 以及量化操作。

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.

昇腾通过 `MIX_AIC`、`MIX_AIV`、`AI_CORE` 和 `AI_VECTOR_CORE` 暴露出可分别调度的立方（cube）单元与向量（vector）单元路径。每个事件都有任务类型，因此时长统计本身就是一种划分。混合类型仍然涉及超出单一物理单元的工作；其名称并不意味着把 kernel 的经过时间直接换算成仅立方单元或仅向量单元的实测利用率。



Source: SemiAnalysis

来源：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.

在一种一阶测算场景下，假定将 Cube 吞吐量翻倍而保持设备的其他计算工作不变，可带来 1.32 倍的阶跃式提升。该测算保持实测的流重叠程度恒定，并保持空闲时间不变。这是一种合成推演场景，而非对新芯片的性能预测：混合任务类型并非排他性的利用率测量指标，且带宽、依赖关系和调度机制都可能改变最终结果。

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.

量化融合同样蕴含优化空间，其吸引力在于存在反复出现的“生产者-消费者”边界，以及在兼顾融合通信的 Vector 耗时中占据了较大比例。

Ecosystem gaps

生态差距

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

然而，潜在的问题在于，这种合作模式或许对 DeepSeek 极为有效，但未必能让昇腾对其他开发者而言同样好用。

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

对 DeepSeek 而言，构建编译器、维护底层算子（kernel）以及调试厂商专有行为，会在一定程度上分散模型研究和产品开发的精力，尽管这可能是暂时的。

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:

对其他所有人而言，一旦硬件配置与 DeepSeek 的环境不一致，就可能出现问题。大部分优化都是针对超节点进行的，而在截至 `vLLM-Ascend_0.26.0rc1` 的版本中，昇腾在 x86 主机上运行仍存在诸多问题：

- No DeepGEMM support on x86 hosts
- x86 主机不支持 DeepGEMM

- Dependency on `memfabric_hybrid` containing hardcoded ARM headers for vLLM-Ascend, blocking startup; vLLM-Ascend doesn't work out of the box.
- 依赖的 `memfabric_hybrid` 中包含针对 vLLM-Ascend 硬编码的 ARM 头文件，导致启动受阻；vLLM-Ascend 无法开箱即用
- CPU-binding skipped for non-ARM CPUs on vLLM-Ascend
- 在 vLLM-Ascend 上对非 ARM CPU 跳过 CPU 绑定
- No NPU + x86 CI, patches needed for some recipes
- 缺乏 NPU + x86 的持续集成（CI），部分运行配方需要补丁

Conclusion

结语

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.

华为在部分基础算子上已经拥有极高效率的实现。DeepSeek 则带来了特定于模型的工作负载与工程能力，将该平台的支撑边界拓展至这些基础算子之外；与此同时，华为的硬件架构与原生技术栈又深刻影响了这些实现的构建方式。由此沉淀下来的编译器与内核开发工作，其意义不仅在于当下的性能表现，更在于能为下一代模型及其落地部署节省大量的工程投入。

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.

更广阔的机遇则在于它有望成为中国 AI 产业的工程乘数。其他提供 DeepSeek 服务的基础设施运营商可以直接复用这些算子内核与部署实践，其他实验室也能依托共享的编译器能力开发新模型。若这些使用者能将修复补丁与优化成果持续反哺，这种收益便会形成复利：每一个新团队在起步时，最棘手的大量底层工作都已铺设完毕。



Source: Dwarkesh Podcast

来源：Dwarkesh 播客

On the Dwarkesh Podcast, Jenson 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, Jenson's nightmare is starting to come true.

在 Dwarkesh 播客中，黄仁勋谈到了他对芯片出口管制的担忧。从目前的软件成果以及在 950PR 上的测试来看，在华为已有工作的基础上加上 DeepSeek 的助力，生态层面的差距确实正在收窄，黄仁勋的噩梦正开始成为现实。