超越计算机：诺思机

一种面向 AI 时代的模型原生架构

JohnRothan*

2026 年 6 月

摘要

七十年来，计算机一直是一台冯·诺依曼机：一个中央处理器，按字、逐条地经由总线从被动的存储中取出指令与数据，四周环绕着外设。这一契约熬过了从大型机到智能手机的每一次器件换代。大语言模型（LLM）作为一种通用计算基底的到来，把它逼到了崩溃的边缘。基础模型推理的主导代价不在于算术，而在于搬运庞大且基本静态的权重张量与不断增长的键值（KV）缓存；在通用硬件上，自回归解码阶段受**内存带宽**支配，而非浮点吞吐。业界当下的答案——"AI PC"，即一台外挂了神经处理单元（NPU）的冯·诺依曼机——把模型当作又一个被加速的子程序。我们认为这是过渡形态，而非终极形态。

我们主张从硬件层面、围绕模型重新设计机器。我们将其结果命名为**诺思机**（Noeton，源自希腊语 *noesis*，" 理解、理智之行为"），其组织原则称为 **模型原生架构**（Model-Native Architecture,MNA ），作为冯·诺依曼机的后继抽象提出，形成对标：von Neumann machine → Noeton。诺思机 立于五条原则之上：(i) **内存中心**组织——权重与 KV 缓存成为一等公民，由存内计算与解耦池化内存就地服务，而非经由一个核心被流式搬运；(ii) **模型原生计算**——一张张量/注意力数据流织物，其中传统 CPU 被降格为管家式协调者；(iii) **模型即操作系统**——一个 LLM 作为内核，智能体是进程，工具是系统调用，向量库是文件系统；(iv) **强制联网、分层认知**——端侧小模型只提供降级模式下的基础功能，完整能力取自网络化的权重池——因此，一台离线的 诺思机 刻意地" 做不了多少事"；(v) **意图驱动交互**——用户的工作单元是" 针对上下文表达的意图"，而非" 针对数据下达的指令"。我们用既有的硬件与系统研究——内存墙、存内计算、CXL 内存池化、晶圆级集成、确定性数据流处理器、神经形态芯片，以及新兴的"LLM 操作系统"——逐一支撑每条原则，并将它们综合为一套连贯的参考设计，配以基于 roofline 与能耗的论证。我们并不构建 诺思机；这是一篇立场论文。我们把它作为一个值得争论的目标提出，并认真对待其代价：强制联网以能力换取了自主、隐私与公平，我们专辟一节讨论这些风险，而非将其轻轻带过。

*为 arXiv 准备的草稿（主分类：cs.AR；交叉：cs.AI）。欢迎指正。

关键词： 计算机体系结构 · 大语言模型 · 内存墙 · 存内计算 · 内存解耦 · 领域专用架构 · LLM 操作系统 · 端云推理。

1 引言

我们今天使用的计算机，在骨子里仍是冯·诺依曼及其合作者于 1945 年所描述的那台机器：一个中央处理单元，从单一的被动存储中按字、逐条地取出指令与操作数，四周环绕着输入/输出外设。这一抽象异常持久。它熬过了从真空管到晶体管再到集成电路的更替，熬过了从大型机到小型机、个人电脑、再到智能手机的演进。缓存、流水线、乱序执行、多核与 GPU，全都是这一契约之*内*的精细化，而非对它的背离。我们仍在为一台存储程序计算机编程；我们仍把*计算*视为稀缺资源，把内存视为在两次计算之间存放东西的地方。

两件事同时发生了变化。第一，这一契约长盛不衰的历史引擎——由 Dennard 缩放带来的、单线程性能稳定而" 免费" 的提升——已经停止。Dennard 缩放的终结与" 暗硅" 的出现 [1] 意味着我们再也无法点亮所能制造的全部晶体管，学界的回应是转向*领域专用架构*：与某一特定负载协同设计的硅，而非被要求包打天下的通用硅 [2]。第二，也更具决定性的是，单一负载已重要到值得为之设计一台机器。Transformer [3] 以及建立其上的大语言模型 [4, 5] 已成为一种*通用基底*：一个学习得到的函数，能起草代码、回答问题、驱动智能体、操控工具，并日益成为人与机器交互的中介。对于计算机越来越大比例的任务而言，模型*就是*应用。

把这两个事实放在一起，一个显而易见的问题随之而来：如果我们将要建造领域专用的机器，而主导领域如今已是基础模型，那么一台*原生地*为基础模型而设计的机器是什么样子？业界已有一个临时答案摆在货架上——"AI PC" [6, 7]：一台普通的冯·诺依曼笔记本，在 CPU 与 GPU 旁加上一颗神经处理单元（NPU）。我们将论证（第 3 节）这是一种过渡形态。它加速了模型，却仍把模型当作客人——又一个由 CPU 经由一套为另一个时代的工作集而设计的内存层级所派发的子程序。更深层的错配被原封不动地留下了。

错配所在。 基础模型推理的决定性代价不是算术，而是数据搬运。一个现代 LLM 是数十至数百 GB 的权重，每生成一个 token 都必须读取一遍；再加上一份随对话增长、且每一步都要重读的键值（KV）缓存。结果是，推理的生成（解码）阶段是*内存带宽受限*的：算术单元空等操作数 [8, 9]。这是一个老问题——" 内存墙" [10]——的当代面孔，而且只有愈演愈烈：峰值算力的增长远快于内存带宽，因此对 LLM 而言瓶颈如今已牢牢落在账本的内存一侧 [11]。一次 roofline 分析 [12, 13] 会把自回归解码置于机器算力天花板之下很远处，紧贴其带宽天花板。冯·诺依曼组织——计算居中、数据从边缘被搬入——对这种负载恰恰是错误的形状。

本文。 我们认真对待这个问题，并用一套设计而非一组基准来回答它。我们提出一台从硬件层面围绕模型组织的机器。由于它身上能被辨认为存储程序意义上的" 计算机" 的东西如此之少——中心的稀缺资源是内存而非计算；内核是一个学习得到的模型而非手写的执行体；其原生工作单元是意图而非指令——我们给它另起一个名字。我们称之为 诺思机 ，源自希腊语 *noesis*，即理解或理智之行为；并称其组织学说为*模型原生架构*（MNA ）。我们希望这个名字与即将到来的时代之关系，正

如” 冯·诺依曼机” 与正在落幕的时代之关系：

$$\text{von Neumann machine} \longrightarrow \text{诺思机}.$$

贡献。

1. **一个诊断**（第 2 节、第 3 节）：我们经由内存墙与 LLM 推理的 roofline 论证，外挂式”AI PC” 恰恰继承了它本应消除的瓶颈，因而不是 AI 时代机器的终极形态。
2. **五条设计原则**（第 4 节）：内存中心组织、模型原生计算、模型即操作系统、强制联网的分层认知、意图驱动交互。
3. **一套参考架构**（第 5 节）：诺思机 , 一个五层栈——内存中心基底、张量原生计算织物、Model-OS 运行时、含端-边-云分层认知的强制网络织物、意图接口——每一层都用既有的硬件与系统研究来实例化，而非凭空臆想。
4. **一份定量动机**（第 6 节）：用 roofline 与数据搬运能耗论证内存中心化的翻转为何取胜，并明确给出假设。
5. **一次诚实的核算**（第 8 节）：明确该设计的代价——强制联网把一台个人机器变成网络化的依赖，对隐私、自主、公平、安全与能耗都有一阶影响，我们将其作为首要问题而非脚注来对待；并讨论（第 9 节）为何这一造物值得一个新名字。

我们想说清认识论上的定位。这是一篇立场与愿景论文，体裁同 [2] 与 LLM-OS 类提案 [14]；它不构建硬件，也不报告任何 诺思机 的实测，因为尚无 诺思机 。它的主张是：这些拼图碎片已存在于文献之中，它们指向一致的方向，而为这一方向*命名并组装*本身就是有用的工作。

2 背景与动机

我们的论证基于三个来自既有文献的观察：冯·诺依曼组织存在一个已知数十年的结构性瓶颈；这个瓶颈对 LLM 推理而言是主导代价而非次要代价；以及学界已独立发展出逃离它所需的大多数硬件要素，只是尚未将它们组装成一台连贯的机器。我们逐一回顾。

2.1 冯·诺依曼瓶颈与内存墙

这一结构性批判与该架构的成熟一样古老。Backus 在其 1977 年图灵奖讲座中命名了” 冯·诺依曼瓶颈”：处理器与存储之间那条唯一的通道，整个计算都必须按字逐一从中泵过；他论证这不仅是性能限制，更是一种*智识*限制，塑造了我们能够*如何*去思考程序 [15]。同一问题的性能面孔由 Wulf 与 McKee 于 1995 年量化为” 内存墙” [10]：由于处理器速度的提升远快于内存延迟，平均访存时间终将主导执行时间，无论处理器变得多快。他们的算术很简单，结论却令人不安——墙不是” 会不会” 的问题，而是” 何时” 的问题。

其间的数十年里，深层缓存层级钝化了这堵墙，而缓存奏效的前提，恰恰是程序复用了能装进快速内存的数据。然而其底层搬运的能耗代价从未消失：把一个字从 DRAM 搬出，其能耗比消费

它的那次算术运算高出约两到三个数量级 [16]。对于 *算术强度*高——每取一字节做许多次运算——的负载，这尚可容忍，因为取数被摊薄了。麻烦在于，如下文所示，LLM 推理恰恰具有相反的性质。

2.2 LLM 推理是内存受限的

一个部署中的基础模型，其时间花在两个阶段。在*预填充*（prefill）期，它吸收提示词，可并行处理所有 token；该阶段算术强度高，受算力限制。在*解码*（decode）期，它逐个 token 地生成回答，每生成一个新 token 都需读取*整个*权重矩阵一次、以及*整个*不断增长的 KV 缓存一次，只为产出一列激活 [9]。因此解码的算术强度极低——在批大小为一时，约为每读一个权重字节做一次乘加——该阶段受内存带宽而非算术单元限制，算术单元在空等操作数 [8]。这不是调优细节，而是服务 LLM 的核心经济事实，也正因如此，大量系统工作专门针对 KV 缓存（如多查询与分组查询注意力通过缩小缓存来恰好缓解这一带宽压力 [8]）。

roofline 模型 [12] 把情形可视化：以可达性能对算术强度作图，一台机器有一条斜的带宽天花板和一条平的算力天花板，二者相交于" 脊点"。脊点左侧的负载受带宽限制，右侧的受算力限制。近期工作把 LLM 解码置于通用加速器脊点的左侧很远处 [13]：我们为之付费的昂贵浮点硅大多空闲，被权重与缓存能多快送达所卡住。我们在 图 4（第 6 节）中示意性地重现这一图景。对本文而言关键的一点是*架构性*的：当约束在于把数据送达计算单元，一个把计算置于中心、数据置于外围的组织，优化的是错误的对象。

2.3 瓶颈只会愈发扩大

人们或许希望这堵墙正在退去。它没有。Gholami 等人 [11] 追踪了相关增长率，发现近几代加速器里，峰值硬件算力（FLOP/s）的增长远快于 DRAM 带宽或互连带宽。因此，算术单元所能消费的与内存所能供应的之间的差距，随每一代而*扩大*，而非缩小。随着模型继续扩张、上下文窗口继续拉长——二者都由经验性的规模律与涌现能力文献所充分激励 [5, 17]（其强读法存在争议 [18]，但无论如何诠释，系统层面的压力都是真实的）——每 token 需搬运的工作集随之增大，错配也随之加深。一台为这一负载的下一个十年而设计的机器，不能继承冯·诺依曼的数据搬运假设却指望它能扩展。

2.4 逃离的要素已然齐备

图景中令人鼓舞的另一半是：过去十年里，计算机体系结构已造出建造一台内存中心、模型原生机器所需的大多数部件——但它们是彼此分立的研究计划，而非一个整合的整体。*存内计算*（PIM）把计算搬到数据所在之处，从根源上攻击数据搬运代价 [19, 20]，而商用 HBM-PIM [21] 证明这是可量产的硅而非仿真。*内存解耦*与基于 CXL 的内存池化 [22, 23] 让内存容量得以独立于计算来分配——这正是一台稀缺资源为权重容量的机器所需要的。*晶圆级集成* [24] 把整个模型常驻于片上 SRAM，从而彻底绕开片外带宽。*确定性数据流处理器* [25, 26] 与最初的张量加速器 [27] 则展示了当计算被塑造成贴合张量负载、而非贴合通用代码时是什么样子。而系统社区已经开始把模型本身当作一个操作系统 [28–30]。这些每一个都是同一台机器的碎片。本文的贡献，是论证它们是*同一台*机器的碎片，并把它画出来。

3 为何"AI PC" 不是终极形态

设立一个稻草人是不公平的。当前这一代具备 AI 能力的个人机器是真实而有用的工程进步，任何诚实的提案都必须精确说明它在何处不足，而非将其一笔抹杀。本节即做此事。

3.1 AI PC 究竟是什么

如今出货的"AI PC"，是一台冯·诺依曼计算机，在 CPU 与 GPU 旁加上了第三个计算引擎：一颗专为端侧推理的矩阵与卷积核而优化的神经处理单元（NPU），并以" 每秒万亿次运算" 作为卖点 [6]。苹果的变体则把一个端侧模型与一个更大的服务器模型配对，当本地模型不足时上升到云端 [7, 31]。两者都是真实的改进：它们把有用的推理带到设备上，降低了小模型的延迟，也把部分数据留在本地。我们并不主张它们一无所成。我们主张的是，它们把机器的重心原封不动地留在了原处。

3.2 三处被继承的错配

1. CPU 仍主宰着机器。 在 AI PC 中，NPU 是一个*设备*：CPU 运行操作系统、调度任务、掌管内存映射，并像派发给任何加速器那样把推理作业派发给 NPU。模型是被宿主调用的客人。但对于一台主要工作就是运行模型的机器而言，这颠倒了自然的层级。模型不是操作系统偶尔调用的子程序；它是、或本应是操作系统*为之存在*的对象。把一个通用顺序核保留为主人、把模型保留为被调用者，等于保留了一套为另一个世界——主导负载是多分支、顺序的应用代码的世界——而建的控制结构，而对越来越大比例的使用而言，那个世界已不再描述机器正在做的事。

2. 内存层级仍以计算为中心。 NPU 经由与其他一切相同的内存层级取得操作数：权重每生成一个 token 都从 DRAM、经过总线、流入加速器。这一外挂为一个负载增添了算术吞吐——更多 FLOP/s——而我们在 第 2.2 节已确立，该负载的约束*不是* FLOP/s 而是带宽。给一个带宽受限的负载增添算力，抬高的是它从未触及的算力天花板，而真正卡住它的带宽天花板纹丝不动。因此 AI PC 完整地继承了内存墙；它仅仅是让空闲的算术单元变得更快。这正是我们视其为过渡形态的最重要理由：它加速的是本不构成瓶颈的那部分。

3. 联网是事后补救，而非契约。 AI PC 把网络当作一条可选的上升路径：能在本地做的就在本地做，必要时再够向云端 [7]。这是一个合理的工程折衷，但它是旧契约*之内*的折衷——在旧契约中，个人电脑是一台自足的机器，它*可以*使用网络。我们将论证（第 4 节、第 5.4 节）AI 时代的机器更应反过来理解：作为一种本质上网络化的认知资源的、薄薄的本地化身，对它而言离线运行是一种刻意降级的模式，而非默认状态。AI PC 没有做出这一选择；它让本地机器保持首要地位，并把认知像其他一切那样外挂在旁边。

3.3 规律：在契约之内做加速

每一处错配都是同一件事的实例。AI PC 在冯·诺依曼契约*之内*加速模型，而非围绕模型重建机器。这是面对一种新主导负载的、历史上正常的首次反应——图形在 GPU 重塑整座数据中心之前，先得到了一颗外挂 GPU；我们应当预期推理会沿同样的弧线前行——但它是首次反应，而非最

终反应。本文的论点是：最终反应将翻转这一关系——模型不再是计算机所加速的设备，而成为机器为之而建的组织中心。本文余下部分描述那台机器。

4 模型原生机器的设计原则

在具体描述 诺思机 之前，我们先陈述生成它的五条原则。每一条都是对某项冯·诺依曼假设的有意翻转，每一条都有既有工作的独立支撑；第 5 节的架构，正是同时认真对待这五条所得到的结果。我们把它们陈述为一份设计宣言，再建造满足它们的机器。

P1 ——内存中心，而非计算中心。 *稀缺且起组织作用的资源是内存，架构应让权重与 KV 缓存成为一等公民。*冯·诺依曼机把内存视为被动存储，中央处理器从中汲取操作数。诺思机 把模型的参数与工作缓存视为机器的主要居民，由被搬到数据处的计算*就地*服务——存内计算 [19, 21]——并由池化、解耦的内存来扩容 [22]，而非经由一个核心被流式搬运。算力密度跟随内存，而非相反。其依据是 第 2.2 节：当负载受带宽限制时，最小化数据搬运就是全部要义。

P2 ——模型原生计算；CPU 被降格。 *主导计算织物是一张张量/注意力数据流引擎；一个传统顺序核仅作为管家式协调者而幸存。*AI PC 保留通用 CPU 为主、模型为客（第 3 节），诺思机 则翻转两者的角色。张量与注意力算子是机器的原生指令，运行在确定性数据流织物上 [25–27]；一个小型 CPU 处理引导、I/O 胶水，以及不属于模型的控制流。顺序核并不消失——总得有人料理家务——但它不再是中心。

P3 ——模型即操作系统。 *一个 LLM 是内核；智能体是进程，工具是系统调用，向量库是文件系统。*诺思机 面向用户的运行时，不是一个偶尔调用模型的手写执行体；它就是模型，作为特权解释器去调度工作、管理上下文、调用工具、并中介 I/O。这一重构在系统研究中已然展开 [14, 28–30]；我们将其采纳为一项硬件设计约束，因为一台内核是模型的机器，理应为该模型提供内核所需的特权、内存抽象与保护域。

P4 ——强制联网；认知是分层的。 *完整能力栖于网络之中；设备只承载一个小模型，在离线时提供降级模式下的基础功能。*这是最具争议的一条，我们刻意陈述之。诺思机 不是一台可以使用网络的自足计算机；它是一种网络化认知资源的本地端点。一个端侧小模型供给一层能力地板——基础听写、简单查询、本地控制——使机器在断连时不至于变成一块砖；但构成完整能力的大模型与池化权重内存，要经由网络去够取，推理在端、边、云之间切分 [7, 32]。这是在一个新层次上对那条老论题——” 网络即计算机”[33]——的回归。我们把这一选择的自主、隐私与公平代价当作首要问题，并在 第 8 节直面之；此处我们只主张：该设计 *做出*了这一选择，而非把联网留作事后补救。

P5 ——意图驱动，而非指令驱动。 *工作单元是” 针对上下文表达的意图”，而非” 针对数据下达的指令”。*冯·诺依曼编程模型向机器呈上一串对显式寻址的数据所作的精确操作序列。诺思机 的原生接口是意图——一个以自然语言或其他模态陈述的目标——在机器所维护的弥漫式上下文中被求解。” 程序” 日益是权重；” 总线流量” 日益是 token；用户的工作是说明*什么*，模型的工作是定夺*如何*。

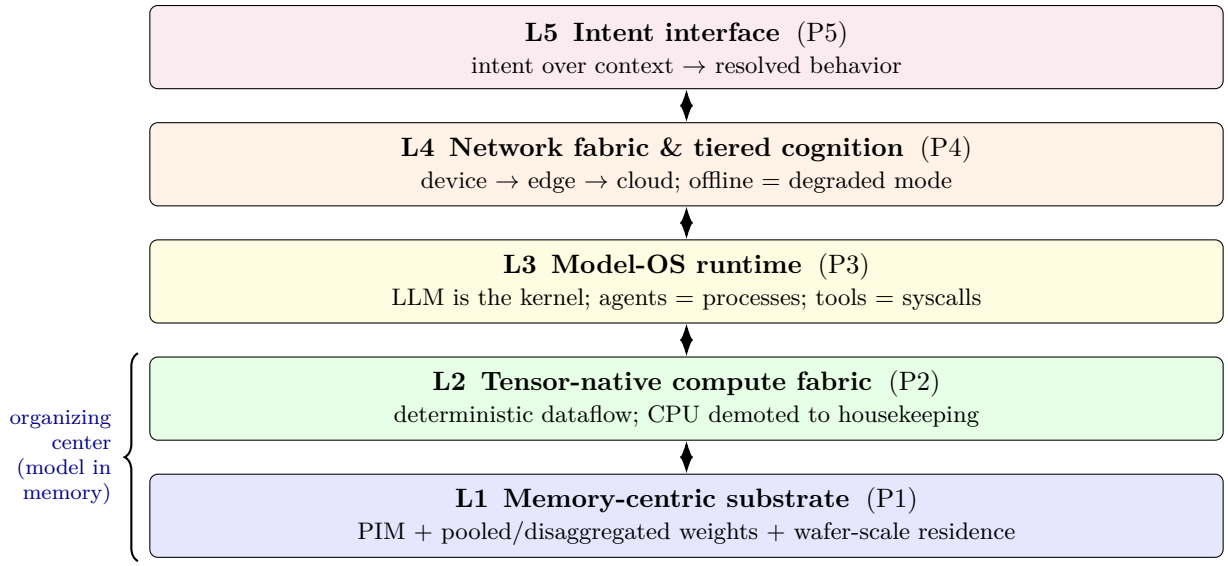

图 1: 诺思机 五层栈。与冯·诺依曼机不同，其组织中心是内存常驻的模型（底部两层），而非一个中央处理器。每一层都标注了实例化它的既有技术。

这条原则塑造的是接口与指令集抽象（第 5.5 节），而非直接塑造硅，但它正是让其余四条凝聚为一台人真正会用的机器的那一条。

这五条并非彼此独立的愿望；它们相互强化。内存中心组织（P1）使模型即内核（P3）变得可负担，因为内核的状态——其权重与上下文——恰是内存系统所要持有之物。分层认知（P4）使一台小设备得以呈现一个极大模型的接口，而这正是意图驱动交互（P5）隐含的承诺。而降格 CPU（P2）是那项结构性变更，它阻止旧契约悄然复辟。下一节将它们组装起来。

5 诺思机架构

现在我们把五条原则组装为一套具体的参考设计。诺思机 组织为五层，如 图 1 所示，并在 图 2 中与冯·诺依曼组织对照。自底向上：一个*内存中心基底*持有并服务模型；一张*张量原生计算织物*执行其算子；一个 Model-OS *运行时*充当内核；一张*网络织物*把机器延伸为分层的端-边-云认知；一个 *意图接口*把它呈现给用户。我们逐层描述，并指明实例化每一层的既有技术，以使该设计成为对已被验证之部件的综合，而非一纸愿望清单。

5.1 第 1 层：内存中心基底

机器的地基是一个被设计来*持有一个模型并就地对其计算*的内存系统，它落实原则 P1。它融合了三种今天分居于不同产品中的既有技术。

存内计算（PIM）把算术嵌入内存的存储体，使推理中受带宽限制的部分无需将权重搬离内存芯片即可执行。通用 PIM 是真实、已出货的硅——UPMEM 的 DRAM 常驻处理器 [20] 与三星的 HBM-PIM，在 HBM 存储体层级嵌入 MAC/SIMD 单元，其内部带宽远高于外部引脚 [21]——而近期工作展示了对 LLM 的具体收益：AttAcc 把受带宽限制的注意力层卸载到 PIM，同时把受算力

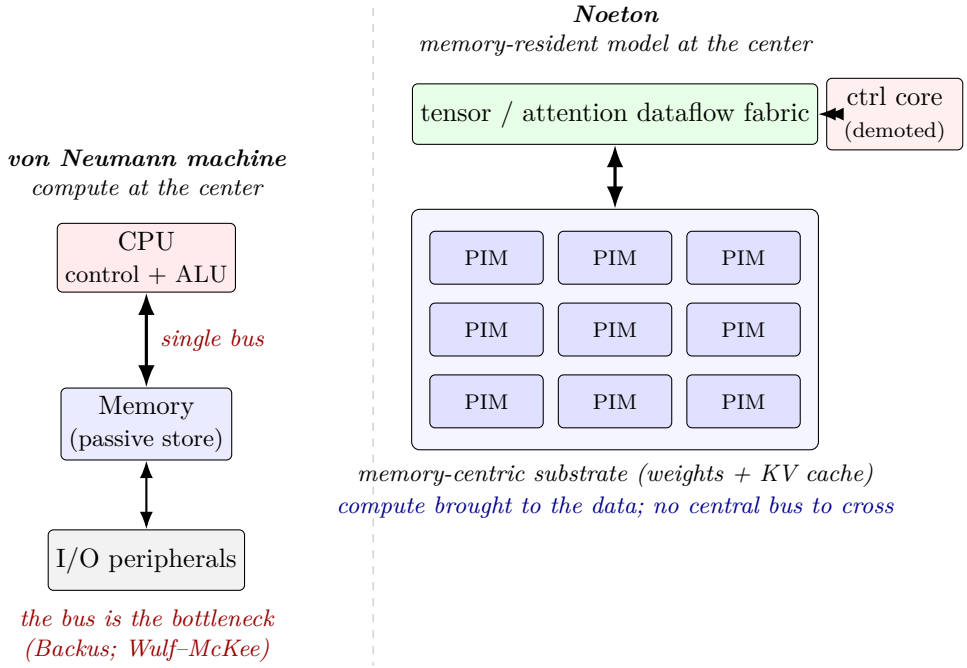

图 2: 冯·诺依曼机（左）对 诺思机（右）。冯·诺依曼机让所有数据经由总线穿过一个中央处理器——这正是 Backus [15] 与 Wulf–McKee [10] 所命名的瓶颈。诺思机 翻转了这幅图：计算被分布进一个内存常驻的模型基底，一个被降格的控制核居于边缘。

限制的前馈层留在传统加速器上，在 1750 亿参数模型上报告了大幅的性能与能效提升 [34]。诺思机 将其推广：受带宽限制的算子（对 KV 缓存的注意力，以及解码中权重流式的矩阵-向量乘）是 PIM 原生的，在数据所在之处执行。为追求超低能耗推理，基底可纳入模拟存内交叉阵列，已有工作表明，经硬件感知训练，它们在 Transformer 类网络上能在器件非理想性下恢复全精度准确率 [35, 36]。

*解耦、池化的权重内存*提供容量。一台稀缺资源为参数容量的机器，想要独立于计算来扩展内存，而这正是基于 CXL 的内存池化所交付的 [22, 23]，并辅以跨层放置页面的操作系统支持 [37]。在 诺思机 中，模型权重栖于一个池化层，设备按需从中汲取：一个小模型常驻本地，更大的模型从边缘或云端的权重池调入。这是 P4 分层认知的硬件基底。

*晶圆级片上常驻*是同一思想的高端极端：把整个模型保持在片上 SRAM，使片外带宽根本不构成约束。Cerebras 晶圆级引擎为训练与推理证明了这一点，并以权重流式的扩展方式把权重存储与计算基底解耦 [24, 38, 39]。诺思机 把晶圆级常驻与 CXL 池化视为同一容量谱的两端：同一架构，按” 多少模型常驻、多少分页” 来参数化。

5.2 第 2 层：张量原生计算织物

基底之上，是执行模型算子的计算织物，它落实原则 P2。其原生” 指令” 是张量与注意力原语，而非标量算术。这一谱系已确立有据：TPU 的脉动矩阵引擎，通过把硅塑造成贴合稠密矩阵乘，展示了一个数量级的每瓦性能优势 [27]；注意力专用加速器如 A^3 [40] 与 SpAtten [41]，通过在硬件中直接利用注意力的结构与稀疏，取得进一步收益；确定性数据流处理器——Groq 的张量流式处理

器 [25] 与 SambaNova 的可重构数据流单元 [26]——则表明，以编译期调度的数据流取代反应式缓存与仲裁器，可获得可预测的低延迟推理，这对一台交互式认知机器至关重要。

诺思机 的织物是一个确定性、静态调度的数据流阵列，与其下的 PIM 基底紧耦合：受算力限制的算子（预填充矩阵乘、前馈层）以高算术强度运行于织物之上，而受带宽限制的算子（解码期的注意力与权重流式）被下推进基底（P1）。这一切分，是把 AttAcc 的洞见 [34] 提升为一条架构原则：*把每个算子放在其约束资源最便宜之处*。一个传统顺序核仍留在芯片上，但处于 P2 所言的降格角色——它引导机器、驱动外设、运行那些确实多分支且顺序的薄控制逻辑，而模型的实际工作发生在织物-基底这对组合上。

5.3 第 3 层：Model-OS 运行时

诺思机 的内核是一个模型，它落实原则 P3。特权运行时不是一个调度进程、把模型当服务调用的手写执行体，而*就是*一个 LLM，去解释意图、规划、调用工具、并管理自己的上下文。这是把"LLM OS" 的构想落到实处 [28]：模型是内核进程；上下文窗口像虚拟内存那样被管理，相关材料在快速工作上下文与较慢的外部存储之间换入换出，恰如 MemGPT 所示 [29]；智能体是内核调度的进程；工具调用是模型够向外部世界的系统调用 [14, 30]。表 2 （第 7 节）给出经典 OS 概念与其 Model-OS 对应物之间的完整对照。

让模型成为内核，会向下层提出它们正为之而建的硬件要求。内核的状态——其权重与活跃上下文——庞大，且必须以高带宽服务，这正是第 1 层为何以内存为中心。内核必须能把其有效内存扩展到常驻之外，这正是第 1 层为何要解耦与分页。内核还必须在智能体-进程之间实施保护、并中介它们对工具的访问，这意味着 诺思机 需要在*模型上下文与工具能力*之上、而非仅在内存页之上定义的硬件保护域——这是我们在 第 8 节标记的一个研究问题，而非宣称已解决。

5.4 第 4 层：网络织物与分层认知

第四层使机器在构造上即是网络化的，它落实原则 P4，如 图 3 所示。认知跨三个尺度分层。在设备上，一个小模型提供一层有保证的能力地板与最低延迟路径。在边缘，一个中型模型与一个区域权重池以低延迟服务常见情形。在云端，最大的模型与完整的权重池提供最高能力。一个请求在能满足它的最便宜层上被求解，而单次推理可 *跨层切分*——例如把受算力限制的预填充与受内存限制的解码分开，置于配置不同的硬件上，正如 Splitwise 在数据中心内所做 [32]，也如端-边-云切分推理方案跨网络所做 [7, 31]。

这一层的决定性承诺是*降级模式*。当 诺思机 离线时，它回退到端侧小模型：基础听写、本地搜索、简单控制与起草仍然可用，但定义这台机器的完整认知能力则不可用。这是对个人电脑契约的一次刻意翻转。我们清楚" 你的电脑断网就几乎不工作" 读起来像一个缺陷；第 8 节论证它其实是对一种依赖的诚实暴露，而 AI PC 的可选上升模型只是把这种依赖隐藏起来；我们也直面其真实代价。这一层的智识祖先是 Sun 在 1980 年代的论题" 网络即计算机"[33]，其最近的出货亲戚是云优先的瘦客户端；诺思机 的不同之处在于，栖于网络中的不是文件或应用，而是*认知*本身。

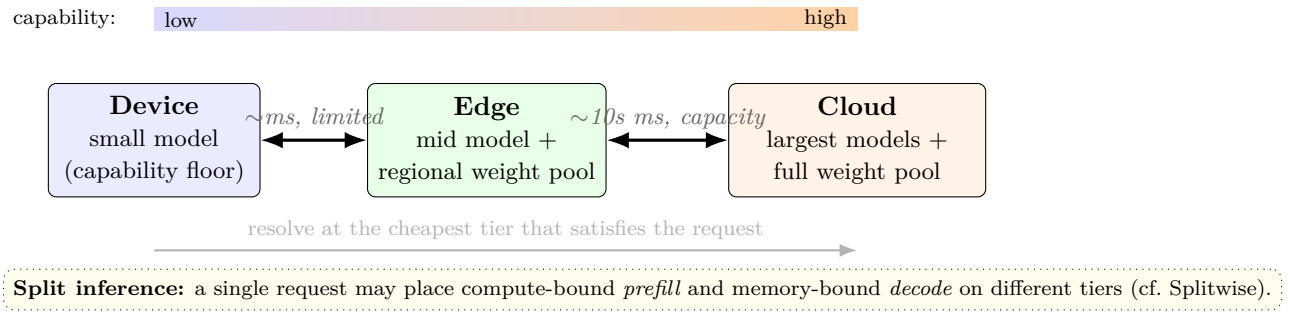

图 3: 分层认知（P4）。一个请求在能满足它的最便宜层上被求解；单次推理可跨层切分（如预填充对解码）。端侧小模型定义了离线的*降级模式*：断连时机器仍可用，但刻意受限。

5.5 第 5 层：意图接口与指令抽象

顶层把机器呈现给用户，并定义其指令集抽象，它落实原则 P5。经典接口向机器呈上一条针对寻址数据的指令流；诺思机 呈上的是一个针对弥漫式上下文的*意图*。用户陈述一个目标——以语言、语音或其他模态——Model-OS 内核在其所维护的上下文中求解之，把它分解为织物上的算子、工具调用，以及必要时跨网络层的上升。在这一抽象中，经典机器的各类比物发生了位移：“程序”主要是模型权重；“指令流”是一串 token；“寻址”是对向量库的检索，而非对线性内存的索引。机器上安装的智能体是它的应用，由内核作为进程调度（表 2）。这一层的贡献不是一套新的控件工具箱，而是对” 既然指令集已是一个模型，那么机器的指令集是什么？ ” 这一问题的连贯回答——即意图进、被求解的行为出，而下面四层之存在，正是为了让这一求解变得快、变得大、变得网络化。

6 定量动机

诺思机 的论据并不只靠文字。本节给出一个一阶的、量级估算的论证——明确是近似的、带有清晰假设——以说明为何内存中心的翻转对 LLM 负载是正确之举。我们并不声称对一台尚不存在的机器做了周期精确的评测；我们只声称，这一量级推理无歧义地指向同一个方向。

6.1 LLM 推理的 roofline

考虑从一个稠密 Transformer 解码单个 token：参数量为 P，每参数 b 字节，批大小为 B。为产出一个 token，机器须把全部 P 个参数读取一次，即传输 $P{\cdot}b$ 字节，并对批中每个序列做约 $2P$ 次浮点运算，即总计 $2PB$ 次。于是算术强度——每字节权重流量上的运算数——为

$$I_{\text{decode}} \approx \frac{2PB}{Pb} = \frac{2B}{b}.$$

惊人之处在于，I_{decode} 与模型大小 P 无关：把模型做大，会搬运更多字节、也按比例做更多算术，强度保持不变。对小批（B 约为一）、16 位权重（$b = 2$）的交互式服务，I_{decode} 约为每字节一次运算。而当代加速器的脊点强度（峰值 FLOP/s 除以峰值带宽）约为每字节 10^2 次运算。因此解码落

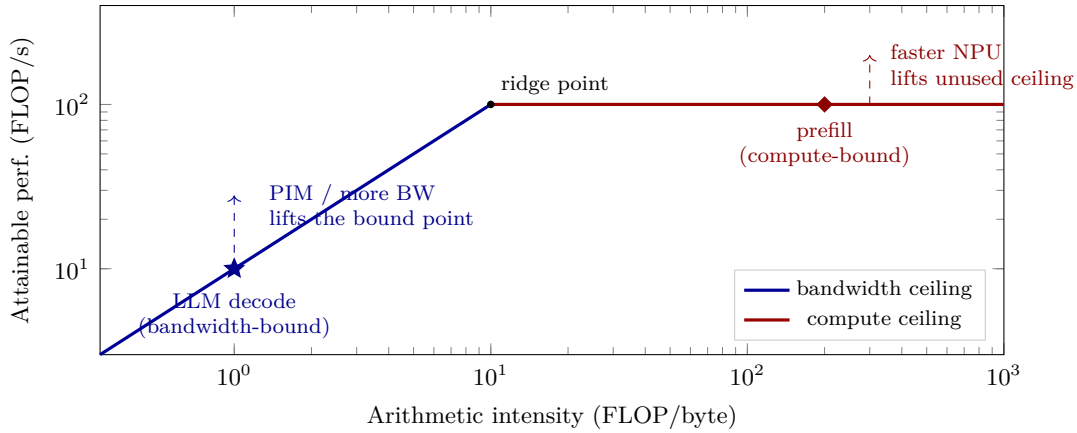

图 4: Roofline 示意。预填充（高算术强度）受算力限制，位于脊点右侧。小批解码被钉在*左侧*一到两个数量级处，受带宽限制：机器为之付费的算力天花板触不可及。抬高算力天花板（一颗更快的 NPU）不会把解码点抬上去；抬高带宽天花板、或经由 PIM 减少每 token 的搬运字节，才会。

在脊点*左侧*一到两个数量级处：深陷带宽受限区，可达性能完全由带宽决定，昂贵的算术单元大多空闲。图 4 示意性地展示了这一点；其定量版本由 [9, 13] 详细展开。

两点直接后果随之而来。第一，*增添 FLOP/s 无助于解码*。抬高算力天花板——AI PC 的外挂 NPU——抬起的是负载从不触及的天花板。只有抬高有效带宽、或减少每 token 必须搬运的字节，才能抬起解码点。第二，确实有效的两个杠杆，恰是 诺思机 的基底（P1）：存内计算通过在权重所在之处计算，减少跨越任何外部边界的字节 [21, 34]；而片上或封装内常驻（晶圆级、HBM）抬高了卡住该点的带宽 [24]。在这一精确意义上，该架构是从 *roofline 推*导而来，而非被断言。

6.2 能耗账本

带宽论证有一个能耗的孪生。该负载的主导能耗代价不是算术，而是把操作数搬向算术：一次 DRAM 访问的能耗，约为它所喂给的那次乘加的 10^2 至 10^3 倍 [16]。由于解码每 token 都要读取整个参数集，每 token 能耗由权重 *搬运*主导；而一台为每个用户的每个 token 都把权重跨板搬运的机器，是在持续地支付这一溢价。把计算搬向数据——PIM 与片上常驻——直接攻击这一主导项，这正是为何内存中心设计报告的是能效提升、而不仅是性能提升 [34, 42]。在车队规模上，这不是一个微架构上的细枝末节：主导一个已部署模型生命周期能耗的是推理而非训练，而 LLM 服务已占数据中心需求中可观且日增的份额 [43–45]。一个减少每 token 搬运字节的架构，就减少每 token 的焦耳，而二者在一阶上是同一个量。

6.3 该分析说明了什么、没说明什么

我们对这一论证的边界很审慎。它说明了主导负载的*约束*在于带宽与数据搬运能耗，并说明 诺思机 的内存中心基底恰好针对这一约束，而外挂加速器针对的是一个不构成约束的对象。它*没有*给出某台 诺思机 的具体加速比或能效数字，因为那取决于一台无人建过之机器的参数——PIM 到织物的带宽、常驻对分页的比例、各网络层的延迟。这些恰是一次未来评测会去测量的量。这一量级估算确立的是*方向*：对于一个受内存限制、被数据搬运主导、且只会增长的负载，正确的架构之举是把内存置于中心、并把计算搬向它。第 5 节中的一切，都是这一结论的下游。

表 1: 诺思机 相对四个参照架构的定位。诺思机 的独特之处不在任何单独一列，而在于把” 内存中心、模型即内核、强制联网” 的组合作为其*组织性*承诺、而非外挂来采纳。

	冯·诺依曼	GPU+CPU 服务器	AI PC	神经形态	诺思机（本文）
内存模型	被动存储, 单总线	深缓存层级 + HBM	同上 + NPU 便笺存储	内存融入神经元	内存中心: PIM + 池化/解耦权重
计算范式	顺序标量核	SIMT 吞吐 + 标量宿主	CPU + GPU + NPU 设备	事件驱动脉冲	张量/注意力数据流；CPU 降格为管家
“操作系统”	手写执行体	CPU 上的 OS; 模型是服务	CPU 上的 OS; 模型被加速	无（定函数）	Model-OS：LLM 即内核, 智能体即进程
网络假设	可选	可选（集群织物）	可选地上升到云	通常独立	**强制**; 离线是降级模式
目标负载	通用顺序代码	训练 + 批量推理	端侧小模型推理	稀疏脉冲/感知	基础模型推理与智能体
约束资源	内存延迟	内存带宽	内存带宽（被继承）	每脉冲能耗	带宽——由 PIM 从源头攻击

7 对比与定位

为精确定位 诺思机 ，我们沿六个维度，将它与四个参照点对比：经典冯·诺依曼机；今天训练并服务大多数模型的常规 GPU+CPU 服务器；第 3 节的 AI PC；以及神经形态架构——后者与 诺思机 共享内存-计算融合，却追逐一种不同的负载。表 1 作了总结。

从表中可见三点。第一，诺思机 *并非*唯一融合内存与计算的架构——神经形态芯片也如此 [42, 46, 47]——但它是为一种不同的负载而融合。神经形态设计面向稀疏、事件驱动的脉冲计算，擅长低功耗感知；诺思机 面向稠密的基础模型推理。它们共享诊断（冯·诺依曼的数据搬运是大敌），却不共享处方。我们视它们为表亲，一台 诺思机 完全可以在其降级模式地板之内承载一座神经形态之岛，用于常开感知。

第二，GPU+CPU 服务器已部分走在通往 诺思机 的路上：它受带宽限制，用 HBM 与之抗衡，且大型部署会解耦内存并切分推理阶段 [22, 32]。诺思机 在一定程度上是在论证：这些数据中心级的适配——内存解耦、阶段切分、PIM 卸载——不仅是服务优化，更是*架构*的预演，理应被下沉进机器的定义，而非外挂到一台本质上冯·诺依曼的服务器上。

第三，AI PC 在除” 目标负载” 外的每一行都与 诺思机 不同，而这正是 第 3 节的要点：它瞄准了对的负载，却用了错的组织，继承了它本应消除的约束资源（带宽）。

7.1 操作系统对照

由于原则 P3（模型即操作系统）最易被误读为比喻，我们把它具体化。表 2 把经典操作系统概念映射到其 Model-OS 对应物；每一行都对应一条既有的系统工作，已在表中引用。这一映射不是为增色而设的类比——它是一份规格说明，规定了 Model-OS 内核必须提供什么，并由此规定了下层必须提供什么硬件支持（在上下文与工具能力之上的保护域、快速的上下文分页）。

表 2: 经典 OS 概念与其 Model-OS 对应物（P3）。每一行都点名一个具体的系统研究实例。该映射同时是一份硬件需求清单：内核需要快速的上下文分页，以及在上下文与工具能力之上定义的保护域。

经典 OS	Model-OS 对应物	实例 / 参考
内核 / 特权执行体	LLM 本身	LLM OS 构想 [28]
进程	智能体	智能体即应用 [14, 30]
系统调用	工具 / 函数调用	工具使用 [30]
虚拟内存 / 分页	上下文窗口管理	MemGPT 上下文分页 [29]
文件系统	向量库 / 检索索引	检索即寻址 [28]
调度器	智能体 / 任务编排	AIOS 内核调度 [30]
设备驱动	工具适配器 / 连接器	工具适配器 [14]
内存保护	上下文与能力隔离	开放问题（第 8 节）

8 含义、风险与开放挑战

一篇只兜售愿景的愿景论文是宣传。诺思机 做了一个确实昂贵的选择——强制联网——而上文的设计中也有若干尚未解决的难题。我们在此直面它们，因为它们既是反对该提案的最有力论据，也是相关工作最重要的方向。

8.1 强制联网的代价

原则 P4 刻意使机器在完整能力上依赖网络。这买来了核心的技术收益——一台小设备得以呈现一个极大模型的接口，由它本地永远装不下的池化内存来服务——但它也卖出了真实的东西，我们应当点名它们，而非让抽象凭省略赢得论证。

隐私与监控。 若完整认知发生在网络上，那么用户的意图——在一台认知机器上，它远比磁盘上的文件更暴露隐私——便会穿越并可能被远端基础设施处理。相对一台自足的 PC，这是监控面的结构性增大。诚实的回应不是否认它，而是针对它来设计：用于推理的机密计算飞地、客户端上下文加密、在降级模式地板之内对敏感意图的端侧处理、以及可验证的删除。苹果的私有云计算方向正是对这种信任结构的一次早期尝试 [7]；这类保证能否做得足够强、足够可审计以证成 P4，在我们看来，是整个提案最重要的开放问题。

自主与" 断网变砖" 问题。 一台断连时几乎做不了事的机器，是一台其主人更不自主的机器。我们在 第 5.4 节论证过，AI PC 的可选上升模型只是*隐藏*了这种依赖——它最有用的能力本就在离线时降级或消失——而 诺思机 至少把这种依赖显式化，并设计了一层有保证的地板（端侧小模型）。但显式不等于可接受。降级模式地板的大小是一个带有政治内涵的政策选择：一台 诺思机 离线时须有多强，这种依赖才算可容忍？我们不认为存在唯一正确的答案，也认为这个问题值得被公开追问，而非由厂商默默定夺。

中心化与锁定。 若认知由网络中的池化权重来服务，那么谁拥有这些权重池，谁就掌握了机器能思考什么的咽喉。这是一种远比今日应用商店动态更集中的依赖，它招致同样的失败模式——抽租、封禁、反复无常的政策——施加于认知本身。缓解之策部分是技术的（开放权重池、联邦与点对点服务、自建一层的权利），部分是治理的；我们不假装技术之策能独力胜任。

公平与数字鸿沟。 一台完整能力需要良好连接的机器，会拉大连接良好者与其余所有人之间的差距，并把它输出：同一台设备，在有光纤的城市里是一件强大的认知工具，在没有光纤的地区则是一件受限的工具。降级模式地板，再一次是相关的杠杆——离线小模型越强，鸿沟越小——这是把端侧能力当作一层应被抬高的地板、而非一个可被容忍的权宜之计的理由。

8.2 尚未解决的技术问题

除联网之外，该架构还留下真实的难题。

上下文与能力之上的保护。 经典机器在进程之间保护内存页。一个 Model-OS 必须在智能体-进程之间保护*上下文与工具能力*——防止一个智能体读取另一个的上下文、或调用它不该调用的工具——而我们并不知道为此设的正确硬件原语（表 2 最后一行）。它是内存保护单元在模型即内核情形下的类比物，而它尚不存在。

确定性、安全与一个学习型内核。 一个内核是学习型模型，意味着它非确定，且会以手写执行体不会的方式失败：它会被提示注入、会幻觉出一次工具调用、会被越狱。让一个学习型内核可信到足以持有特权——沙箱化其工具调用、约束其权限、验证其行动——本身就是一项安全议程，而第 2 层的确定性数据流选择 [25] 只触及其中的时序一面。

编程与调试一台意图机器。 当程序是权重、指令流是 token（P5），正确性的经典工具——源码、断点、确定性重放——便不能直接套用。调试一台 诺思机 意味着什么？我们提出这个问题而不予以解决；它很可能与 P1 一样深刻。

软件生态与惯性。 最后，最大的现实障碍不是硅，而是已装机的存量。冯·诺依曼契约是几乎所有现存软件的基底。一台 诺思机 要么模拟它（P2 的降格 CPU 部分就为此而在），要么与它共存一段漫长的过渡。我们预期现实路径是混合的——诺思机 式的子系统在其余仍属常规的机器内生长，恰如 第 7 节的数据中心适配已在做的那样——而非一次干净的替换。名字是一处目的地，而非对某个” 换旗日” 的预言。

9 论命名：为何不再叫计算机

本文通篇用了一个新词，这一选择并非装饰。一个名字是关于某物是何种东西的主张，我们想为如下主张辩护：我们所描述的机器，在种类上已不同到值得拥有一个名字。

9.1 为何需要新名字

” 计算机”（computer），在词源与历史上，是一台*计算*的机器——在存储程序的指挥下，对数据执行算术与逻辑运算。这个词命名的是机器中心的活动。以此标准衡量，第 5 节的造物被误名了。它中心的稀缺资源是内存而非计算（P1）；它的主导织物运行一个学习得到的函数，而非一套手写指令的存储程序（P2、P3）；它的原生工作单元是针对上下文求解的意图，而非施于寻址数据的指令（P5）；它在构造上是网络化的，而非自足的（P4）。当机器的中心不再是存储程序意义上的” 计算”，继续称它为” 计算机” 便是一个范畴错误，悄悄把旧契约重新偷运进来——我们认为，这部分解释了为何 AI PC 被构想成一台被加速的计算机，而非一台不同的机器。

历史先例是真实的。” 冯·诺依曼机” 本身就是一个做了有用工作的名字：它让架构师能就一*类*机器及其特征性瓶颈来推理 [10, 15]，而非就某个具体产品来推理。我们想要一个为 AI 时代做同样工作的名字，使其处于如下关系

$$\text{von Neumann machine} \longrightarrow \text{诺思机},$$

以便人们能说” 这是一个 诺思机 式的设计”，从而一举唤起 P1–P5 这一整套，正如” 这是一个冯·诺依曼式的设计” 唤起对单一存储的取指-译码-执行。

9.2 所提议的名字

我们提议 **诺思机**（Noeton），源自希腊语 *noesis*（诺思），即理解或理智之行为。这个名字把机器组织性活动从*计算*到*认知*的转移推到前台，以一个概念性词根代替” 冯·诺依曼机” 中的个人尊称作平行，且简短、易读、在体系结构文献中尚无人占用。其架构学说——一台 诺思机 所实例化的那套原则——我们称为 *模型原生架构*（MNA ），它作为形容词（” 一个模型原生的设计”）独立于这个专有名词而通行良好。

关于中文译名。 自然的音译” 诺顿机” 与已确立的 Norton（杀毒软件与出版社）中文名相撞，故我们采用 **诺思机**作为推荐中文名：它保留了 *noe-* 的读音，而” 思” 字（思考）承载了与英文造词意图相同的” 认知压倒计算” 之义。我们在英文中统一用 Noeton，在中文中统一用诺思机。

9.3 我们考虑过的备选

我们考虑并否决了若干备选，并把它们记录下来，因为这一选择确实可议。*认知引擎*（Cognitive Engine）透明、无需注解，但” 引擎” 暗示计算机*内部*的一个部件，而非对它的替代，这低估了主张。*模型原生机*（Model-Native Machine, MNM）最具描述性，我们也保留” 模型原生” 作为架构之名，但作为机器名它拗口、读起来像行话。*推理机*、*张量机*、*认知子*（Cognitron）则各自过度承诺于某一层（计算织物，或某种具体的设备样式），而牺牲了整体。我们偏好一个植根于机器定义性*活动*——理解——的名字，而非植根于其机制的名字，理由与” 计算机” 胜过” 算术引擎” 相同：活动之名比机制之名更经得起时间。读者若不喜欢 Noeton，尽可代之以自己的；实质主张是，这一造物在种类上不同，且应有一个名字，而非非此名不可。

10 相关工作

诺思机 是一次综合，其新意在于组装，而非任何单一部件。我们按相关工作所支撑的层次来组织，并在每种情形下说明我们从中取用什么、又在何处与之分道。

领域专用架构作为一项纲领。 我们的框架是 Hennessy 与 Patterson 之号召的直系后裔——他们呼吁一个与负载协同设计的领域专用架构的新黄金时代 [2]，其动因正是 Dennard 缩放的终结与暗硅 [1]。该纲领泛泛地主张 DSA，而我们主张基础模型如今已是*那个*值得为之设计整机的领域，并把协同设计推得更远——越过加速器，进入操作系统与接口。

内存中心硬件。 第 5.1 节的基底立于存内计算 [19–21]、注意力专用 PIM 卸载 [34]、模拟存内计算 [35, 36]、CXL 内存池化与解耦 [22, 23, 37]、以及晶圆级集成 [24, 38, 39]，并辅以已把推理本身当作跨 GPU、CPU、磁盘的内存层级卸载问题来处理的软件 [48]。它们每一个都把内存中心性当作在常规系统*之内*应用的技术；我们与之不同之处，是把它当作机器的*组织原则*，其余的围绕它排布。

张量与数据流加速器。 计算织物取材于脉动张量引擎 [27]、注意力加速器 [40, 41]、以及确定性数据流处理器 [25, 26, 49]。它们通常被定位为*服务于*一个宿主的加速器；我们把织物定位为机器的主计算，而把宿主 CPU 当作配件（P2）。

冯·诺依曼批判及其替代。 我们立足其上的结构性批判很古老 [10, 15]，而神经形态计算是逃离它的最坚决的既有尝试 [42, 46, 47, 50]。我们共享其诊断，而在目标负载上有别（第 7 节）：稠密的基础模型推理，而非稀疏脉冲计算。

模型即操作系统。 第 3 层采纳了迅速发展的"LLM OS" 脉络——Karpathy 的构想 [28]、MemGPT 受 OS 启发的上下文分页 [29]、以及 AIOS 内核与 AIOS-智能体生态 [14, 30]。这些工作几乎全是*软件*，运行在常规硬件上。我们的贡献，是追问一个模型即内核要求什么硬件，并把这一要求当作其下各层的设计驱动。体裁上最近的先例是 AIOS-智能体愿景论文 [14]；它在精神上相近、在范围上相反，因为它设想的是软件生态，而我们重构其下的机器。

网络化与切分推理。 第 4 层立于阶段切分服务 [32]、端云混合基础模型 [7, 31]、以及自" 网络即计算机" [33] 一路传下的漫长瘦客户端谱系之上。这些系统工作把切分推理当作一种服务优化；我们把强制的、分层的认知当作机器的定义性属性，并附以一个显式的降级模式。

负载与经济学。 最后，动机植根于 Transformer 及其能力由规模驱动的特征 [3–5, 17]（涌现的强读法存在争议 [18]），植根于 LLM 推理的 roofline 分析 [9, 12, 13]、数据搬运能耗鸿沟 [16]、以及大规模服务日增的能耗代价 [43–45]。它们确立了该负载是庞大、增长、受带宽限制、且高耗能的——整套设计正由这四个事实而出。

11 产业轨迹（2024–2026）：起草以来的证据

一篇立场论文应当能被它所描述的市场所证伪。在本文起草前后的时间窗口里，产业朝着五条原则中的每一条都发生了可度量的移动；本节记录其中最清晰的数据点，以及它们迫使我们对自身主张做出的两处细化。

走向内存中心组织（P1）。 以 KV 缓存为中心重组整个生产集群的推理系统 Mooncake——其组织核心是缓存而非算力——获得 FAST 2025 最佳论文奖，并已并入主流 vLLM 软件栈 [51]。器件侧，两大 DRAM 厂商启动了 LPDDR6-PIM 的联合标准化 [52]，CXL 3.1 内存扩展控制器进入客户送样阶段。最引人注目的是华为 CloudMatrix 384 超节点：NPU、CPU、DRAM 与 SSD 经统一总线直接互连而*无需 CPU 中介*，其聚合内存容量约为同级 GPU 机柜的 3.6×、带宽约 2.1× [53] ——机柜级的池化、对等寻址内存已不再是假说。

走向强度匹配的计算（P2）与切分推理（P4）。 最有力的单点确证来自在位者本身：NVIDIA 的 Rubin CPX 是一颗*只为预填充而造*的 GPU——高算术吞吐刻意搭配廉价的 GDDR7 而非 HBM——其配备 HBM 的同门则服务带宽受限的解码阶段，Dynamo 框架在两个资源池之间调度请求 [54, 55]。当最没有动机去拆分 GPU 的厂商开始出货按阶段特化的硅片时，「预填充与解码不该挤在同一块芯片上」就已从立场变成了产品。

走向模型操作系统（P3）与意图接口（P5）。 模型上下文协议（MCP）从单一厂商发布（2024 年 11 月）到被所有主要模型提供商采用、再到移交 Linux 基金会中立治理，只用了十三个月 [56]——这是计算史上「系统调用面」最快的一次标准化，其驱动力正是当年催生 POSIX 的同一种 $N \times M$ 适配爆炸。苹果将端侧模型与 Private Cloud Compute [57] 配对部署，把与本文分层认知原则一致的两级降级设计装进了数亿台消费设备。

两处细化。 证据也在两处修正了我们。其一，*近存*先于*存内*到来：定制 HBM 基底裸片、CXL 池化与超节点织物的商业化快于 bank 级 PIM，因此 L1 基底应被读作一个演进序列（近存池化 → bank 级存内计算），而非一步到位。其二，KV 缓存压缩（多头潜在注意力、稀疏注意力）减缓的是缓存*容量*的增长；它并不能把解码从 roofline 的带宽受限斜坡上移走——恰恰相反，把 token 预算花在解码上的测试时计算负载，正在提高推理中属于这道斜坡的份额。

这些进展没有一项与本文协同发生；但它们全部弯向本文所指的方向。我们将此视为一个证据：用 von Neumann → 诺思机 来解读当下，至少是看清产业既定走向的一种富有成效的方式。

12 结论

七十年来，我们建造冯·诺依曼机并称之为计算机，因为其中心是对存储程序的计算。我们已论证，新兴时代的主导负载——基础模型推理——并不契合这一中心。它受权重与缓存的搬运所限，而非受算术所限；它的自然内核是一个学习得到的模型，而非手写的执行体；它的自然工作单元是意

图，而非指令；而它的完整能力，日益是从网络中够取之物，而非藏于一只盒子之内。业界的首次回应 AI PC，在旧契约之内加速模型，并继承了它本应消除的那个瓶颈。

我们提议认真对待另一条路：从硬件层面、围绕模型重新设计机器。其结果——由模型原生架构组织的 诺思机 ——立于五条原则之上（内存中心组织、模型原生计算、模型即操作系统、强制联网的分层认知、意图驱动交互），并把既有、已被验证的技术——存内计算、解耦内存、晶圆级集成、确定性数据流织物，以及新兴的 LLM OS——组装为一台连贯的机器，而 roofline 与能耗账本都表明它指向正确的方向。我们刻意为它命名，因为当一台机器的中心是理解而非计算，旧名字便会把旧假设偷偷带回。

我们对其代价直言不讳。强制联网以能力换取了自主、隐私与公平，而该架构若干最难的问题——上下文与能力之上的保护、学习型内核的安全、调试一台意图机器之义——仍然开放。我们视它们不是击沉提案的反对，而是该提案所催生的研究议程。

一份研究议程。 若方向无误，接下来具体的问题是：

1. *基底。*对给定设备类别，常驻（晶圆级/HBM）与分页（CXL/网络）权重内存之间的正确平衡为何？第 5.2 节的算子放置切分又要求什么样的 PIM-到-织物接口？
2. *内核。*上下文与工具能力的硬件保护原语为何（表 2 缺失的那一行）？一个学习型内核又如何被约束到足以安全持有特权？
3. *网络。*降级模式地板须有多大，P4 的依赖才算可接受？而能证成强制联网的隐私保证，又能否做得足够强、足够可审计？
4. *评测。*当 诺思机 与 AI PC 对能力栖于何处做出不同假设时，对二者的一次公平基准究竟测量什么？

我们不期待一个” 换旗日”。现实之路，正是数据中心里已然可见的那一条——内存解耦、阶段切分、PIM 卸载正一次一个优化地，把服务器悄然变成某种更像 诺思机 的东西。我们的贡献，是为这些优化所走向的目的地命名，论证它是一处连贯之地而非偶然，并把抵达那里的代价与收益一并摆上桌面。无论 诺思机 一词能否留存，我们都相信它所命名的机器值得为之建造——也值得趁其契约仍在书写之时、现在就争论。

致谢

本文为愿景论文，未构建或评测任何新系统。感谢计算机体系结构与机器学习系统社区的广泛工作——本文综合并立足于这些工作之上。

参考文献

[1] Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. Dark silicon and the end of multicore scaling. In *Proc. 38th Annual International Symposium on Computer Architecture (ISCA)*, pages 365–376, 2011. doi: 10.1145/2000064.2000108.

[2] John L. Hennessy and David A. Patterson. A new golden age for computer architecture. *Communications of the ACM*, 62(2):48–60, 2019. doi: 10.1145/3282307.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. arXiv:1706.03762.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1877–1901, 2020. GPT-3; arXiv:2005.14165.

[5] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[6] Microsoft. Introducing Copilot+ PCs. Microsoft Official Blog, May 20, 2024, 2024. https://blogs.microsoft.com/blog/2024/05/20/introducing-copilot-pcs/.

[7] Apple Machine Learning Research. Introducing Apple's on-device and server foundation models, 2024. https://machinelearning.apple.com/research/introducing-apple-foundation-models.

[8] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

[9] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In *Proceedings of Machine Learning and Systems (MLSys)*, 2023. arXiv:2211.05102.

[10] Wm. A. Wulf and Sally A. McKee. Hitting the memory wall: Implications of the obvious. *ACM SIGARCH Computer Architecture News*, 23(1):20–24, 1995. doi: 10.1145/216585.216588.

[11] Amir Gholami, Zhewei Yao, Sehoon Kim, Coleman Hooper, Michael W. Mahoney, and Kurt Keutzer. AI and memory wall. *IEEE Micro*, 44(3):33–39, 2024. doi: 10.1109/MM.2024.3373763. arXiv:2403.14123.

[12] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: An insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, 2009. doi: 10.1145/1498765.1498785.

[13] Zhihang Yuan, Yuzhang Shang, Yang Zhou, Zhen Dong, Zhe Zhou, Chenhao Xue, Bingzhe Wu, et al. LLM inference unveiled: Survey and roofline model insights. *arXiv preprint arXiv:2402.16363*, 2024.

[14] Yingqiang Ge, Yujie Ren, Wenyue Hua, Shuyuan Xu, Juntao Tan, and Yongfeng Zhang. LLM as OS, agents as apps: Envisioning AIOS, agents and the AIOS-agent ecosystem. *arXiv preprint arXiv:2312.03815*, 2023.

[15] John Backus. Can programming be liberated from the von Neumann style? a functional style and its algebra of programs. *Communications of the ACM*, 21(8):613–641, 1978. doi: 10.1145/359576.359579. 1977 ACM Turing Award Lecture.

[16] Mark Horowitz. 1.1 Computing's energy problem (and what we can do about it). In *IEEE International Solid-State Circuits Conference (ISSCC)*, pages 10–14, 2014. doi: 10.1109/ISSCC.2014.6757323.

[17] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, et al. Emergent abilities of large language models. *Transactions on Machine Learning Research (TMLR)*, 2022. arXiv:2206.07682.

[18] Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2304.15004.

[19] Onur Mutlu, Saugata Ghose, Juan Gómez-Luna, and Rachata Ausavarungnirun. A modern primer on processing in memory. *Emerging Computing: From Devices to Systems (Springer); arXiv:2012.03112*, 2022.

[20] Juan Gómez-Luna, Izzat El Hajj, Ivan Fernandez, Christina Giannoula, Geraldo F. Oliveira, and Onur Mutlu. Benchmarking a new paradigm: An experimental analysis of a real processing-in-memory architecture. *IEEE Access*, 10:52565–52608, 2022. doi: 10.1109/ACCESS.2022.3174101. PrIM benchmarks, UPMEM; arXiv:2105.03814.

[21] Young-Cheon Kwon, Suk Han Lee, Jaehoon Lee, et al. Aquabolt-XL: Samsung HBM2-PIM with in-memory processing for ML accelerators and beyond. In *IEEE Hot Chips 33 Symposium (HCS)*, 2021. doi: 10.1109/HCS52781.2021.9567191.

[22] Huaicheng Li, Daniel S. Berger, Lisa Hsu, Daniel Ernst, Pantea Zardoshti, Stanko Novakovic, Monish Shah, Samir Rajadnya, Scott Lee, et al. Pond: CXL-based memory pooling systems for cloud platforms. In *Proc. 28th ACM Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2023. doi: 10.1145/3575693.3578835. Distinguished Paper; arXiv:2203.00241.

[23] Hasan Al Maruf and Mosharaf Chowdhury. Memory disaggregation: Advances and open challenges. *ACM SIGOPS Operating Systems Review*, 57(1), 2023. doi: 10.1145/3606557.3606562. arXiv:2305.03943.

[24] Sean Lie. Cerebras architecture deep dive: First look inside the hardware/software co-design for deep learning. *IEEE Micro*, 43(3):18–30, 2023. doi: 10.1109/MM.2023.3256384.

[25] Dennis Abts, Jonathan Ross, Jonathan Sliwa, Brian Taylor, Chia-Hsin Owen Chen, Gregory Thorson, Igor Tomic, et al. Think fast: A tensor streaming processor (tsp) for accelerating deep learning workloads. In *Proc. 47th Annual Int. Symposium on Computer Architecture (ISCA)*, pages 145–158, 2020. doi: 10.1109/ISCA45697.2020.00023.

[26] Raghu Prabhakar, Ram Sivaramakrishnan, Darshan Gandhi, Yun Du, Mingran Wang, Xiangyu Song, et al. SambaNova SN40L: Scaling the AI memory wall with dataflow and composition of experts. *Proc. 57th IEEE/ACM Int. Symposium on Microarchitecture (MICRO); arXiv:2405.07518*, 2024.

[27] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, et al. In-datacenter performance analysis of a tensor processing unit. In *Proc. 44th Annual Int. Symposium on Computer Architecture (ISCA)*, pages 1–12, 2017. doi: 10.1145/3079856.3080246. arXiv:1704.04760.

[28] Andrej Karpathy. LLM OS: The LLM as the kernel process of a new operating system. https://x.com/karpathy/status/1707437820045062561, 2023. Talk "Intro to Large Language Models" and accompanying remarks; not peer-reviewed.

[29] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

[30] Kai Mei, Zelong Li, Shuyuan Xu, Ruosong Ye, Yingqiang Ge, and Yongfeng Zhang. AIOS: LLM agent operating system. *arXiv preprint arXiv:2403.16971*, 2024.

[31] Apple. Apple intelligence foundation language models. *arXiv preprint arXiv:2507.13575*, 2025.

[32] Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative LLM inference using phase splitting. In *Proc. 51st Annual Int. Symposium on Computer Architecture (ISCA)*, 2024. doi: 10.1109/ISCA59077.2024.00019. arXiv:2311.18677.

[33] John Gage. The network is the computer, 1984. Sun Microsystems slogan; see IEEE Spectrum coverage, https://spectrum.ieee.org/does-repurposing-of-sun-microsystems-slogan-honor-history.

[34] Jaehyun Park, Jaewan Choi, Kwanhee Kyung, Michael Jaemin Kim, Yongsuk Kwon, Nam Sung Kim, and Jung Ho Ahn. AttAcc! unleashing the power of PIM for batched

transformer-based generative model inference. In *Proc. 29th ACM Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2024. doi: 10.1145/3620665.3640422.

[35] Manuel Le Gallo, Riduan Khaddam-Aljameh, Milos Stanisavljevic, et al. Hardware-aware training for large-scale and diverse deep learning inference workloads using in-memory computing-based accelerators. *Nature Communications*, 14:5282, 2023. doi: 10.1038/s41467-023-40770-4.

[36] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R. Stanley Williams, and Vivek Srikumar. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. In *Proc. 43rd Int. Symposium on Computer Architecture (ISCA)*, pages 14–26, 2016. doi: 10.1145/3007787.3001139.

[37] Hasan Al Maruf, Hao Wang, Abhishek Dhanotia, Johannes Weiner, Niket Agarwal, Pallab Bhattacharya, Chris Petersen, Mosharaf Chowdhury, Shobhit Kanaujia, and Prakash Chauhan. TPP: Transparent page placement for CXL-enabled tiered-memory. In *Proc. 28th ACM Int. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2023. doi: 10.1145/3582016.3582063. arXiv:2206.02878.

[38] Sean Lie. Inside the cerebras wafer-scale cluster. *IEEE Micro*, 44(4), 2024. doi: 10.1109/MM.2024.3386628.

[39] Nolan Dey et al. Benchmarking the performance of large language models on the cerebras wafer-scale engine. *arXiv preprint arXiv:2409.00287*, 2024.

[40] Tae Jun Ham, Sung Jun Jung, Seonghak Kim, Young H. Oh, Yeonhong Park, Yoonho Song, Jung-Hun Park, Sanghee Lee, Kyoung Park, et al. A^3: Accelerating attention mechanisms in neural networks with approximation. In *IEEE Int. Symposium on High Performance Computer Architecture (HPCA)*, pages 328–341, 2020. doi: 10.1109/HPCA47549.2020.00035. arXiv:2002.10941.

[41] Hanrui Wang, Zhekai Zhang, and Song Han. SpAtten: Efficient sparse attention architecture with cascade token and head pruning. In *IEEE Int. Symposium on High Performance Computer Architecture (HPCA)*, pages 97–110, 2021. doi: 10.1109/HPCA51647.2021.00018. arXiv:2012.09852.

[42] Dharmendra S. Modha, Filipp Akopyan, Alexander Andreopoulos, Rathinakumar Appuswamy, John V. Arthur, et al. Neural inference at the frontier of energy, space, and time. *Science*, 382 (6668):329–335, 2023. doi: 10.1126/science.adh1174. IBM NorthPole.

[43] Anonymous. How hungry is AI? benchmarking energy, water, and carbon footprint of LLM inference. *arXiv preprint arXiv:2505.09598*, 2025.

[44] Anonymous. Quantifying the energy consumption and carbon emissions of LLM inference via simulations. *arXiv preprint arXiv:2507.11417*, 2025.

[45] Arman Shehabi, Sarah J. Smith, Alex Hubbard, Alexander Newkirk, Nuoa Lei, Md Abu Bakar Siddik, Billie Holecek, Jonathan Koomey, Eric Masanet, and Dale Sartor. 2024 united states data center energy usage report. Technical Report LBNL-2001637, Lawrence Berkeley National Laboratory, 2024.

[46] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014. doi: 10.1126/science.1254642.

[47] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018. doi: 10.1109/MM.2018.112130359.

[48] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Daniel Y. Fu, Zhiqiang Xie, Beidi Chen, Clark Barrett, Joseph E. Gonzalez, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. FlexGen: High-throughput generative inference of large language models with a single GPU. In *Proc. 40th Int. Conference on Machine Learning (ICML)*, 2023. arXiv:2303.06865.

[49] Zhe Jia, Blake Tillman, Marco Maggioni, and Daniele P. Scarpazza. Dissecting the graphcore IPU architecture via microbenchmarking. *arXiv preprint arXiv:1912.03413*, 2019.

[50] Steve B. Furber, Francesco Galluppi, Steve Temple, and Luis A. Plana. The SpiNNaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014. doi: 10.1109/JPROC.2014.2304638.

[51] Ruoyu Qin, Zheming Li, Weiran He, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: Trading more storage for less computation — a KVCache-centric architecture for serving LLM chatbot. In *USENIX Conference on File and Storage Technologies (FAST)*, 2025. Best Paper Award; arXiv:2407.00079.

[52] SK hynix and Samsung Electronics. LPDDR6-PIM standardization initiative for low-power processing-in-memory, 2025. Industry roadmap disclosures, 2025.

[53] Pengfei Zuo et al. Serving large language models on Huawei CloudMatrix384, 2025. arXiv:2506.12708.

[54] NVIDIA. NVIDIA unveils Rubin CPX: A new class of GPU designed for massive-context inference, September 2025. Press release. https://nvidianews.nvidia.com/news/nvidia-unveils-rubin-cpx-a-new-class-of-gpu-designed-for-massive-context-inference.

[55] NVIDIA. NVIDIA Dynamo: A datacenter-scale distributed inference serving framework, March 2025. Announced at GTC 2025. https://github.com/ai-dynamo/dynamo.

[56] Anthropic. Introducing the Model Context Protocol, November 2024. Donated to the Linux Foundation Agentic AI Foundation in Dec. 2025. https://modelcontextprotocol.io.

[57] Apple Security Engineering. Private cloud compute: A new frontier for AI privacy in the cloud, June 2024. https://security.apple.com/blog/private-cloud-compute/.