PowerInfer-2: Fast Large Language Model Inference on a Smartphone

None

2024 年 9 月 2 日

1 摘要

本文介绍了**PowerInfer-2**，一个专为智能手机上的大型语言模型（LLM）推理而设计的高速框架，特别适用于那些模型大小超过设备内存容量的情况。PowerInfer-2的核心思想是利用智能手机中的异构计算、内存和I/O资源，通过将传统的大矩阵计算分解为细粒度的神经元集群计算来优化推理速度。

具体来说，PowerInfer-2采用了一个多态神经元引擎，能够针对LLM推理的不同阶段调整计算策略。此外，它引入了分段神经元缓存和细粒度神经元集群级流水线技术，有效地减少和隐藏由I/O操作带来的开销。

PowerInfer-2的实现和评估显示，其在两款智能手机上支持广泛的LLM模型，推理速度相比于最先进的框架提高了多达29.2倍。值得注意的是，PowerInfer-2是第一个能够在智能手机上以每秒11.68个token的生成速度运行TurboSparse-Mixtral-47B模型的系统。对于完全适合内存的模型，PowerInfer-2可以在保持与llama.cpp和MLC-LLM相当的推理速度的同时，实现约40%的内存使用量减少。更多详细信息和演示视频，请访问项目网站：www.powerinfer.ai/v2。

2 介绍

大型语言模型（LLMs）凭借其出色的理解和生成类人文本的能力，极大地提升了我们的日常生活，并改变了我们的工作环境。当前最先进的LLMs，如GPT-4和Claude-3，通常部署在配备最先进的GPU（例如NVIDIA H100）的数据中心。这些GPU提供了大量的高带宽内存，能够实现数千Tflops的计算能力。同时，越来越多的趋势表明，人们正在将LLMs部署在广泛使用的智能手机上，从而将这些设备转变为智能个人助理。这种转变旨在充分利用丰富的个人数据，同时通过避免将私人数据传输到云服务来保持隐私。

然而，尽管智能手机使用广泛，但在推理LLMs时，由于处理能力有限和内存大小受限，它们难以满足复杂的计算需求。为了解决这些问题，研究人员探索了两种在资源受限条件下进行LLM推理的解决方案。第一种方法是部署缩减版的LLM，例如Google的Gemini Nano 3.25B，它使用不到2GB的内存，但牺牲了部分智能能力，以适应内存限制。另一种方法则是在推理过程中降低LLM权重的计算和存储需求。例如，PowerInfer通过将热激活的神经元分配到GPU，冷神经元分配到CPU，从而在个人电脑上实现了11倍的推理速度提升。然而，这些方法在智能手机上效果不佳，因为手机硬件异构且存储设备带宽较低，无法支持高效的并发访问。

为了解决这些挑战，本文介绍了PowerInfer-2，这是第一个专为智能手机设计的高效LLM推理框架，能够处理高达47亿参数的模型，超过了设备的内存容量。与其前身PowerInfer一样，PowerInfer-2利用LLM推理中的动态稀疏激活特性，即每次推理迭代只需要处理一小部分神经元，而不是整个模型的权重。这种方法显著降低了推理过程中的计算需求，同时提高了局部性，使得PowerInfer-2能够建立一个有效的内存缓存，保持最常使用的神经元在内存中，从而减轻了与I/O操作相关的开销。

与PowerInfer不同，PowerInfer-2的关键挑战在于如何充分利用现代智能手机中的高度异构的计算单元（如big.LITTLE CPU内核、GPU和NPU）。如果推理过程不能充分利用硬件特性，可能会导致次优的生成速度。另一个挑战是不可避免的缓存未命中所导致的I/O开销。尽管PowerInfer-2使用稀疏激活来减少推理过程中所需的权重数量，但它仍然需要大量的I/O读取操作来从存储中获取权重，这可能会对推理性能产生不利影响。

为了解决这些挑战，PowerInfer-2的核心思想是将LLM推理中典型的粗粒度矩阵计算分解为细粒度的

神经元集群计算。神经元集群由多个神经元组成，其数量由XPU的特性、内存和I/O来确定，以充分发挥特定硬件组件的能力。具体来说，为了利用智能手机中的异构XPU，PowerInfer-2设计了一个多态神经元引擎，为LLM推理过程中的预填充和解码阶段提供不同的计算模式。在预填充阶段，所有用户输入序列中的tokens被同时处理，PowerInfer-2将所有神经元合并为一个大集群，以最大化NPU在处理大矩阵计算时的优势。相反，在解码阶段，由于批量大小为1且表现出显著的稀疏性，PowerInfer-2使用小的神经元集群，以利用CPU内核在这种较轻计算任务中的灵活性。

神经元集群粒度进一步允许PowerInfer-2减轻推理过程中I/O开销的影响。PowerInfer-2引入了一个分段缓存，该缓存按神经元粒度操作，并为不同的LLM权重类型设计了特定的缓存策略，有效提高了缓存命中率。此外，为了减少由I/O操作导致的计算延迟，PowerInfer-2提出了一种细粒度的神经元集群级流水线技术，将I/O操作与神经元集群计算重叠起来。这种方法显著减少了I/O延迟相关的等待时间。

我们通过扩展PowerInfer（增加了12,000行代码）实现了PowerInfer-2，并将其部署在两款智能手机（OnePlus 12和Ace 2）上，这两款手机均配备了异构的高通XPU，并分别拥有24GB和16GB的DRAM内存。PowerInfer-2支持多种LLM模型，包括Llama-2（7B, 13B），TurboSparse-Mistral（7B）和TurboSparse-Mixtral（47B）等。我们的评估表明，PowerInfer-2在多个模型和智能手机硬件上表现出色，相比于现有的最先进框架（如LLM in a Flash和llama.cpp），实现了平均3.94倍（最高可达4.38倍）和25.4倍（最高可达29.2倍）的速度提升。特别地，PowerInfer-2是第一个在移动平台上支持TurboSparse-Mixtral-47B模型的系统，生成速度达到每秒11.68个tokens，比llama.cpp快21.2倍。PowerInfer-2的另一个重要优势是其在模型推理期间减少内存使用的能力。例如，对于较小的模型（如7B大小），PowerInfer-2的技术能够在保持与llama.cpp和MLC-LLM相同推理速度的同时，节省近40%的内存使用。

3 背景与动机

3.1 LLM推理及其指标

LLM推理由两个阶段组成：预填充阶段和解码阶段。在预填充阶段，用户的提示通过LLM在一次迭代中处理，生成第一个token。而在解码阶段，LLM以自回归的方式逐一生成token。预填充阶段生成的token作为生成第二个token的输入，第二个token又作为生成第三个token的输入，这一过程持续进行，直到输出序列完成或到达序列结束（EOS）token。

这两个阶段展示了不同的计算模式，需要优化两个关键指标：在预填充阶段生成第一个token的时间（TTFT）和在解码阶段生成每个token之间的时间（TBT）。预填充阶段处理所有提示token，计算负担较重；相对地，解码阶段每次迭代只处理一个token，因此计算需求较低。因此，LLM推理系统必须针对这些阶段设计专门的计算策略，以有效地优化性能指标。

3.2 可预测的稀疏激活

主流的LLM，如GPT-4和Llama-2，采用解码器专用的Transformer架构。这种架构包括多个Transformer层，每层包含一个注意力块和一个前馈神经网络（FFN）块。注意力块在序列中建立token之间的关系，而FFN块则根据注意力块建立的关系解释和处理这些关系。现代LLM通常采用分组查询注意力（Group Query Attention），这减少了注意力块中权重的数量，使前馈网络（FFN）块占据了总权重的近80%。FFN块中的激活函数（如ReLU家族函数）导致大量的稀疏激活现象：大多数神经元（表示为FFN权重矩阵中的行或列）处于未激活状态，因为它们的计算对最终输出影响最小。

幸运的是，FFN中的神经元激活可以在计算每个FFN块之前进行预测，这已经被之前的研究所探索。例如，PowerInfer和DejaVu使用小型MLP网络来预测每个FFN块的动态神经元激活。通过这些准确的预测器，可以显著减少FFN中的神经元计算数量，从而加速推理过程。

3.3 智能手机存储分析

通常，智能手机没有足够的DRAM内存来容纳整个LLM。因此，模型的一部分权重可能存储在外部存

储器中，例如骁龙8gen3中的通用闪存存储（UFS）4.0。本节分析了智能手机UFS的性能特征，这些特征指导了PowerInfer-2的I/O设计。

3.3.1 读取吞吐量与块大小

首先，我们评估了UFS 4.0的随机和顺序读取吞吐量。一个显著特点是UFS的读取带宽随读取块大小而变化。通常，无论是顺序读取还是随机读取，块越大，读取带宽越高。例如，当块大小设置为512KB时，顺序和随机读取的带宽都达到最大值，分别为4 GB/s和3.5 GB/s。而当块大小减少到4KB时，带宽达到最低值，随机读取带宽为450 MB/s。

3.3.2 随机读取与数据范围

UFS的随机读取表现出一个有趣的现象，即随机读取的性能受到读取范围的影响。具体来说，读取范围越小，带宽越高。如图1b所示，在UFS 4.0中，如果4KB随机读取范围设置为128MB、256MB和512MB，那么128MB范围的带宽最高，达到1 GB/s，而512MB范围的带宽最低，低于850 MB/s。因此，在128MB范围内，4KB随机读取的带宽超过了8KB和12KB块大小的读取带宽。

3.3.3 读取吞吐量与CPU内核

第三个观察是，读取带宽受发出读取命令的CPU频率影响。更高频率的CPU内核与更高的读取带宽相关联。例如，当使用3.3GHz频率的大核心进行4KB随机读取时，带宽达到1 GB/s。相反，当使用2.2GHz频率的小核心进行相同的随机读取时，带宽仅为760 MB/s。这种相关性是因为发出读取命令的CPU内核需要运行UFS驱动程序，因此更高的频率能够更快地处理与UFS相关的I/O操作，包括中断和队列管理。

3.3.4 读取吞吐量与核心数量

最后一个观察是，与NVMe不同，移动设备中的UFS存储只有一个命令队列，本质上缺乏内部并发能力。因此，使用多个核心发出I/O命令不会比使用单个核心提高I/O带宽。如表1所示，使用多个核心进行4KB随机读取甚至会导致I/O性能下降多达40%，这是由于UFS命令队列中的争用造成的。

总结：当需要将一些模型权重存储在移动设备的存储介质上时，一个高效的LLM系统必须充分考虑存储介质的性能特征，以最大限度地提高I/O带宽并最小化与I/O操作相关的性能开销。

4 PowerInfer-2 概述

传统的LLM推理通常依赖于将矩阵计算作为推理的基本单位，这种方法在智能手机的异构硬件环境中引入了大量的计算和I/O开销。这种粗粒度的计算方法无法有效利用XPU的灵活计算能力。更糟的是，如果部分矩阵权重存储在存储设备上，那么在开始矩阵计算之前，必须等待这些权重被加载到内存中，从而导致显著的I/O等待时间。

本文介绍了**PowerInfer-2**，一个专为智能手机设计的高速LLM推理框架。其设计实现了三个目标：（1）低推理延迟：在预填充阶段（TTFT）和解码阶段（TBT）期间最小化推理延迟；（2）低内存占用：在模型大小超过设备内存限制时，仍能实现LLM的低延迟推理；（3）灵活性：确保设计能够无缝适配具有不同计算、内存和存储容量的智能手机。

4.1 神经元集群及其架构

在本文中，我们提出了一种称为神经元集群的计算抽象，专门为异构计算场景下的LLM推理设计。PowerInfer-2在神经元集群的粒度上进行计算和I/O操作，这些集群可以在计算过程中动态组合多个被激活的神经元，其数量由计算单元的计算能力决定。例如，在解码阶段，当计算由CPU内核执行时，分配给每个CPU内核的神经元集群大小小于预填充阶段NPU处理的集群。通过使用这种抽象，PowerInfer-2能够充分利用具有不同计算能力的XPU，有效隐藏I/O开销。

图2展示了PowerInfer-2的总体架构，该架构分为在线（右侧部分）和离线（左侧部分）过程。在线部分在神经元集群粒度上提供推理服务，包括四个协作组件：多态神经元引擎（§4.1），内存中的神经元缓存（§4.2），灵活的神经元加载（§4.3），和神经元集群级别的I/O流水线（§4.4）。

多态神经元引擎使用完全不同的计算模式处理预填充和解码阶段。在预填充阶段，神经元集群包含所有的神经元，主要依赖于NPU处理大规模矩阵乘法的效率。

在解码阶段，引擎使用预测器来识别哪些神经元将在计算之前被激活。然后，计算引擎将这些激活的神经元合并到一个小的神经元集群中，并利用CPU内核动态计算该集群，从而显著减少计算需求和运行时内存使用。

在推理过程中开始计算之前，计算引擎从神经元缓存中检索神经元权重，该缓存经过优化以利用LLM推理中观察到的神经元级访问的局部性。如果发生缓存未命中，PowerInfer-2会发出I/O命令从存储中获取未缓存的神经元权重。为了减少I/O延迟，PowerInfer-2引入了一种新的流水线机制，该机制可以同时处理神经元集群和I/O操作。此外，PowerInfer-2通过自适应地捆绑和加载神经元（由模型的量化决定）来最小化I/O开销。

为了自动适应不同的模型或智能手机，在每个新模型首次在新手机上服务之前，会进行离线过程。在在线推理开始之前，该过程涉及接收三种类型的输入：模型权重、用户输入和硬件规格。它输出一个执行计划，描述在线推理过程中涉及的每个组件的配置，并指导在线过程。

具体来说，离线规划器输出计算、内存和I/O的配置。对于计算部分，规划器根据计算单元的计算能力，在不同阶段或层次上确定CPU和NPU的使用比例。在内存配置方面，为了在内存使用和推理性能之间取得平衡，规划器使用户能够在运行PowerInfer-2之前设置所需的推理速度。基于该速度设置，PowerInfer-2计算所需的最佳缓存大小。对于I/O配置，规划器触发探查器来测量模型的稀疏性以及热神经元和冷神经元的分布。

5 神经元感知的运行时推理

5.1 多态神经元引擎

PowerInfer-2引入了一种多态神经元引擎，该引擎能够动态地将神经元组合成神经元集群，以利用LLM推理阶段和异构XPU的不同计算特性。

5.1.1 以NPU为中心的预填充阶段

在预填充阶段，所有提示token被同时处理。尽管这些tokens表现出高度的稀疏性并激活不同的神经元，但由于这些激活的聚合，整体稀疏性显著下降。因此，PowerInfer-2在预填充阶段不使用预测器来计算激活的神经元，而是直接将所有神经元合并为一个大集群。由于NPU在处理大规模矩阵乘法方面表现优异，PowerInfer-2在预填充阶段利用NPU进行计算。

虽然CPU内核不参与矩阵计算，但PowerInfer-2在预填充阶段利用CPU执行NPU的必要准备任务。首先，由于内存有限，PowerInfer-2依靠CPU内核在预填充阶段将存储在闪存中的权重加载到内存中。其次，由于当前NPU不支持直接使用量化权重进行计算，PowerInfer-2使用CPU内核在NPU计算前对数据进行去量化处理。

图3-a展示了CPU和NPU如何在Transformer层级上协作，以执行神经元集群粒度的预填充推理。NPU计算需要使用与CPU共享的一小部分内存。因此，在NPU计算开始之前，CPU应提前将需要的矩阵权重加载到共享内存中。在特定LLM层内，在NPU进行任何矩阵乘法之前，多个CPU中核从神经元缓存中读取量化权重并将其去量化为fp16格式，最终将结果存储在CPU和NPU之间的共享内存中。同时，PowerInfer-2使用大核心异步加载下一层的所有矩阵权重到神经元缓存中。通过去量化、中核计算和大核心I/O操作的重叠，减少I/O延迟。

5.1.2 以CPU为中心的解码阶段

与预填充阶段不同，解码阶段在每次迭代中只专注于一个token，表现出显著的稀疏性，因为权重矩阵中只有一小部分神经元（约10

具体来说，PowerInfer-2在解码阶段使用CPU内核计算注意力块和FFN块。虽然注意力块不表现出稀疏性，但当输入只是一个向量时，CPU内核仍然提供较低的计算延迟。对于FFN块，PowerInfer-2最初将FFN块的输入向量传递给预测器，该预测器预测FFN的权重矩阵中需要激活的神经元并将它们合并为一个神经元集群。每个CPU内核然后获取一个集群，并计算集群内神经元和输入向量之间的结果。

图3-b展示了不同CPU内核进行的解码阶段推理。CPU内核首先从神经元缓存中读取注意力块的权重并与输入向量计算。然后，运行预测器来确定后续权重矩阵中神经元的激活状态。在第三步中，CPU内核将激活的神经元分配到几个集群中，每个内核负责计算其集群内激活的神经元与输入向量的结果，并最终在一个屏障处聚合结果。如果这些神经元在神经元缓存中，CPU内核将与输入向量一起计算它们。如果发生缓存未命中，运行在CPU内核上的I/O线程会异步将神经元加载到缓存中，并最终通知计算线程完成计算。

5.2 内存中的神经元缓存

高效的缓存设计可以显著减少昂贵的存储I/O活动，从而优化端到端的推理性能。传统的LLM推理方法依赖于生成每个token所需的长链条神经元的遍历，这不具备局部性，使任何缓存设计无效。为此，PowerInfer-2引入了一种经典的双队列策略来实现其LRU神经元缓存，该缓存以单个神经元的粒度管理LLM缓存。该系统维护两个双向链接列表队列，分别标记为活动和非活动，神经元在这两个队列中的顺序由其最近访问的时间决定，最近访问的神经元位于队列头部。

在运行时，所有神经元最初加入非活动队列。每当重新访问时，它们会被提升到活动队列的前端。已经在活动队列中的神经元则移动到下一个间隔中。当缓存达到容量时，当活动队列的占用率达到缓存空间的90

6 执行计划的生成

如今的智能手机配备了各种硬件规格，例如不同的CPU能力、I/O吞吐量和DRAM大小。在这些设备上部署LLM的用户也有不同的目标。一些用户可能会在生成速度和内存使用之间寻求平衡，而另一些用户则希望最大化硬件利用率以提高速度。此外，模型本身在权重数量、结构和稀疏性水平方面也存在差异。为了解决这些复杂性，PowerInfer-2 包含了一个专门设计的离线规划器，以制定最优执行计划来满足这些不同的需求。

6.1 执行计划

表2中标记为Var的所有符号表示执行计划的输出，描述了运行时推理组件的配置。$c_{compute}$ 表示分配用于计算的CPU内核集合，而 c_{io} 指定了用于I/O操作的CPU内核。变量n表示选择的CPU内核总数，包括计算和I/O内核。内存限制 m 设置了运行时神经元缓存的大小。最后，规划器输出执行计划的估计生成速度 s。

6.2 输入参数

表2 还列出了三类输入参数：

- **硬件**：从硬件中获取的参数，例如CPU FLOPS、I/O吞吐量和内存带宽。
- **用户**：用户指定的参数，例如CPU限制、内存限制和解码速度的下限。
- **模型**：关于模型的信息，由离线分析器收集，如模型大小、稀疏性水平和缓存特性等。

为了准确测量硬件和模型特性，规划器利用一个离线分析器来确定它们的具体参数。对于硬件，分析器会执行一系列微基准测试来评估各个组件的性能，包括CPU功能 $F_{cpu}(i)$，I/O 带宽 $B_{io}(i,s)$ 和内存带宽 B_{memory}。对于模型参数，分析器首先静态测量神经元的大小和数量。然后，通过在通用数据集上运行模型（从Wikipedia和RefinedWeb中抽样，涉及多达1000万个token），收集诸如神经元热度 $P_{activated}$ 和缓存未命中率 $P_{miss}(m)$ 等动态数据。此外，分析器还在不同的内存限制下测量特定模型的 $P_{miss}(m)$。

6.3 成本模型

在收集到输入参数后，规划器使用成本模型来生成执行计划。目标是最大化生成速度 s，同时满足用户指定的约束。解码速度 s 与解码一个token所需的时间成反比，这由该token的计算时间决定。当我们高效地重叠计算和I/O操作时，就能最大化 s。通过定义目标函数和约束条件，构建的模型可以通过成熟的SMT求解器来解决。在我们的实现中，我们使用Z3求解器来解决成本模型。

$$\text{Maximize} \quad s = 1/T_{decode}$$

$$T_{decode} = T_{attn} + T_{pred} + \max(T_{ffn}, T_{io})$$

为了解码时间，我们首先对计算时间进行建模。计算时间主要受注意力块、预测器和FFN块的影响。计算涉及将这些组件的计算工作负载除以CPU浮点运算能力（FLOPS）。

$$T_{cpu} = T_{attn} + T_{pred} + T_{ffn}$$

$$T_{attn} = C_{attn}/F_{cpu}, \quad T_{pred} = C_{pred}/F_{cpu}, \quad T_{ffn} = C_{ffn} \cdot P_{activated}/F_{cpu}$$

$$F_{cpu} = \sum_{i \in C_{compute}} F_{cpu}(i)$$

随着FFN块计算与神经元加载的重叠，规划器还需要考虑I/O传输时间。这是通过将从闪存传输的神经元体积除以I/O带宽来计算的。传输体积取决于激活率和缓存未命中率。

$$V_{io} = N_{total} \cdot P_{activated} \cdot P_{miss}(m) \cdot S_{neuron}$$

$$T_{io} = V_{io}/B_{io}(c_{io}, S_{neuron})$$

最后，规划器计算从内存中加载神经元的时间，这与在运行时激活的注意力块、预测器和神经元的权重大小有关。内存时间通过将激活神经元的总权重除以内存带宽来确定。

7 实现

PowerInfer-2是在PowerInfer的基础上开发的，后者是一个专为稀疏激活LLM设计的最先进的服务框架。通过集成额外的12K行C++代码，这些改进涵盖了几个关键领域，包括多态神经元引擎、神经元缓存、灵活的神经元加载以及神经元集群级别的I/O流水线。

由于PowerInfer-2依赖于需要root权限的特权系统API（例如锁定内存页的mlock），因此我们在Android平台上构建了它。尽管无需更改系统内核，但在一个有root权限的Android系统上，仍然可以为开发和调试提供相当大的灵活性。此外，PowerInfer-2本质上设计为无需内核修改，因此可以很容易地移植到其他操作系统，如iOS平台。

目前，PowerInfer-2的实现支持各种规模的LLM模型，包括Llama-2系列（7B，13B）、TurboSparse-Mistral（7B）和TurboSparse-Mixtral（47B）。

8 性能评估

在本节中，我们评估了PowerInfer-2在各种模型和智能手机硬件上的性能。

8.1 实验设置

硬件：我们选择了一款高端和一款中端的OnePlus智能手机进行评估，具体配置见表3。选择这些手机的原因有两个：首先，这些硬件配置代表了高端和中端智能手机的典型规格。OnePlus 12是一款配备旗舰硬件的高端智能手机，包括顶级的高通SoC，而OnePlus Ace 2代表了上一代的智能手机。其次，这两款手机都支持root，可以绕过供应商指定的应用程序限制，从而解锁智能手机的全部计算能力。

模型：我们选择了四种不同架构和模型大小的语言模型，分别是TurboSparse-Mistral-7B（简称“Mistral-7B”）、Llama-7B/13B和TurboSparse-Mixtral-47B（简称“Mixtral-47B”）。

基线：我们将PowerInfer-2与三种最先进的LLM推理框架进行比较：llama.cpp、LLM in a Flash（简称LLMFlash）和MLC-LLM。Llama.cpp目前是支持将部分模型权重卸载到闪存存储（通过mmap）的最快大型模型推理框架，同时也支持Ollama等其他框架。LLM in a Flash专为PC环境设计，并且未开源。MLC-LLM使用移动GPU加速大型模型推理，但不支持权重卸载。因此，我们仅将PowerInfer-2与MLC-LLM进行内存充足条件下的比较。

负载：我们从实际LLM任务中选择负载，包括多轮对话、代码生成、数学问题解决和角色扮演，以充分展示PowerInfer-2的效率。

8.2 基于卸载的性能评估

在本节中，我们评估了在可用内存受限的情况下，PowerInfer-2在不同模型上的解码和预填充性能，并与llama.cpp和LLMFlash进行对比。

8.2.1 解码性能

我们首先比较了PowerInfer-2与最先进的LLM推理框架的解码速度。在高端的OnePlus 12上，PowerInfer-2在TurboSparse-Mixtral-47B模型上相较于LLMFlash和llama.cpp平均提速3.94倍（最高达4.38倍）和25.4倍（最高达29.2倍）。在中端的OnePlus Ace 2上，PowerInfer-2的平均提速分别为2.99倍（相较于LLMFlash）和13.3倍（相较于llama.cpp）。

这种提速主要归因于PowerInfer-2在解码过程中使用的高效神经元流水线和灵活的神经元加载策略，有效地减少了I/O操作的开销，并提高了不同模型的整体性能。

8.2.2 预填充性能

在本节中，我们评估了PowerInfer-2在提示长度为128和512个token时的预填充性能。图7显示了PowerInfer-2与其他基线的提示处理速度。在提示长度为128个token时，PowerInfer-2在高端OnePlus 12上相较于LLMFlash快6.95倍，相较于llama.cpp快9.36倍；在中端OnePlus Ace 2上，对于512个token的提示，PowerInfer-2的速度相较于LLMFlash快13.3倍。

这种加速效果主要归因于PowerInfer-2在预填充阶段选择了NPU中心策略。首先，NPU在执行大批量计算时比CPU和GPU更强大；其次，PowerInfer-2利用顺序I/O来提升预填充阶段的效率；最后，通过在当前层计算期间预取下一层的权重，PowerInfer-2有效地将权重加载与计算重叠，从而优化性能。

8.2.3 不同内存容量下的性能

在现实使用场景中，智能手机通常需要同时运行多个应用程序，导致可用内存变化。在这些条件下，为了评估PowerInfer-2的适应性，我们测试了其在7GB到19GB内存配置下的性能。图9展示了TurboSparse-Mixtral-47B模型在不同内存配置下的解码速度。

当可用内存仅为7GB时，PowerInfer-2的解码速度为12.3 tokens/s。通过设计，非FFN层权重（1GB）、预测器权重（2.6GB）和FFN权重的量化尺度（2.7GB）驻留在内存中，剩余300MB用于中间张量、KV缓存和其他运行时内存，总计6.6GB内存使用。在最大内存19GB下，PowerInfer-2的解码速度为11.68 tokens/s，相比LLMFlash和llama.cpp分别快3.12倍和21.2倍。

8.3 内存内性能

如果模型大小完全适合设备内存，PowerInfer-2可以节省内存使用量，同时保持高解码速度。本节评估了在足够内存可用时TurboSparse-Mistral-7B的解码性能。我们将PowerInfer-2与llama.cpp和MLC-LLM进行比较，后者分别使用CPU和GPU进行解码。结果如图11所示。

通过卸载一半的模型权重，PowerInfer-2可以节省大约1.5GB的内存，平均解码速度接近llama.cpp和MLC-LLM两倍。总的来说，PowerInfer-2在保持用户体验的同时，有效减少了已适应内存模型的内存消耗。