BitShare：用于压缩大语言模型权重存储的敏感度感知分箱与比特共享

Qiuyu Ren、Sudipta Paria、Aritra Dasgupta、Swarup Bhunia 电气与计算机工程系（ECE），佛罗里达大学，美国佛罗里达州盖恩斯维尔 32611{qiuyuren、sudiptaparia、aritradasgupta}@ufl.edu, swarup@ece.ufl.edu

arXiv:2611.28607v1 [cs.LG] 21 Nov 2026

摘要—**在批次大小较小时，使用大语言模型进行解码受到内存带宽限制，因此每个权重所存储的比特数在很大程度上决定了其成本。分组整数量化以相同精度存储一个组中的每个权重，尽管组内输入通道的敏感度各不相同。我们提出 BITSHARE，这是一种无码本权重格式，保留分组量化的整数网格和反量化器，但在每个组内采用不同精度。敏感度感知分箱将每个组的通道置于固定位置的三个区间中；静态位共享结合最优位值分配，使敏感度较低的权重共享其低位，每个低位在每组中或每对权重中仅存储一次。GPTQ 的共享感知版本会选择每个存储值，无需微调和码本。BITSHARE 有两种模式。BITSHARE-U 为每个组采用相同布局，因此其解码器通过固定布线传递比特；BITSHARE-N 则在相同比特预算内为每个组选择布局。在 Pile 上校准且存储量相同的情况下，在 WikiText-2 和 C4 上，这两种模式在 3 比特和 2 比特下的困惑度都低于 GPTQ。在 Llama-2 和 Llama- 3 模型的 2 比特设置下，BITSHARE-N 相较于 SliM-LLM 最多可将 WikiText-2 困惑度降低 70%，相较于 GPTAQ 最多降低 35%。在达到 SliM-LLM 的 2 比特困惑度时，BITSHARE-U 每权重所存储的比特数减少 13.6-19.4%，从而将动态随机存取存储器权重流量降低相同比例。在 SkyWater 130nm 中综合实现并经过验证的硬件解码器，以与 INT-n 和 SliM-LLM 解码器相同的吞吐量重构 BITSHARE 权重。**

索引词—**大语言模型、权重量化、混合精度、比特共享、分箱、硬件解码器；**

一、引言

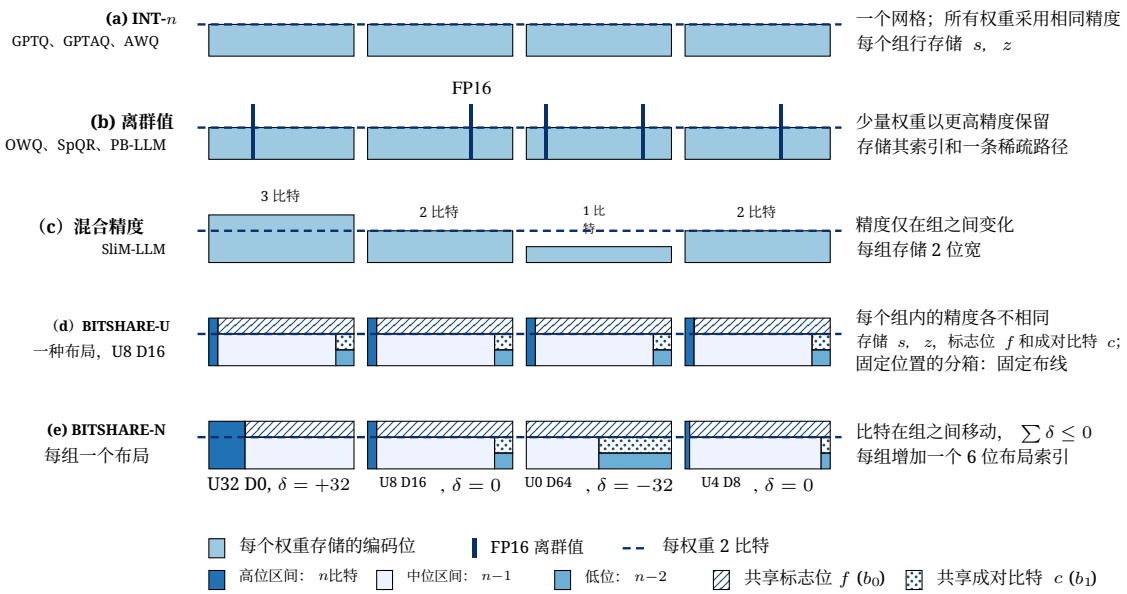
大语言模型的服务受其权重限制。在较小的批次大小下，每生成一个词元都要从内存中读取所有权重，权重流量远超算术运算 [1], ，因此每权重比特数越少，解码速度越快，单个设备可容纳的模型也越大。训练后量化通常以分组整数格式存储权重，将每行划分为若干组，每组包含 128 个输入通道，并为每组设置一个缩放因子、一个零点，以及每个权重对应的一个整数编码。这些格式在 4 比特下几乎无损，但在 3 比特下性能有所下降，2 比特下尤为明显。在 2 比特下，采用调优后裁剪范围的 GPTQ 会使 Llama-3-8B 和 Qwen2.5-0.5B 在 WikiText-2 上的困惑度分别增至原来的 6.4 倍和 9.9 倍。

大多数低比特方法保留这种格式并改进编码器 [2]–[5], ，使组中的每个权重保持相同的精度，尽管一个组的输入通道对层输出的影响极不均衡。离群值方法 [6]–[8] 将少量权重保留为更高精度，但必须存储其位置。SliM-LLM [9] 在不同组之间调整精度，这需要为每个组存储位宽，并会降低解码吞吐量，同时仍将一个组中的所有权重一视同仁。码本和网格格式 [10]–[12] 能够实现更低的 2 比特困惑度，但需要通过查找表或网格进行解码，并且通常需要微调。在一个组内调整精度更加困难，因为为每个权重设置位宽至少需要每个权重 1 个比特，这相当于 2 比特格式预算的一半。

关于位级结构的研究提出了另一条路径。加速器会将一组中的低位比特剪除，并替换为共享常量 [13], ；RRAM设计只存储一次相同的位列 [14], ；纹理格式、测试压缩和总线编码则将多个值绑定到由编码器选定的一个存储比特上（第 II 节）。权重的低位比特承载的信息很少，而在组中敏感度较低的通道里，所承载的信息更少，因此这些权重可以从共享的存储比特中读取低位比特。共享会约束权重，而不是丢弃比特。共享最低位比特的权重仍可取奇偶性相同的任意编码，相当于使用步长加倍、偏移量按组行选定的网格。在 INT3 网格上，这一步长仍比 INT2 的步长细 14%。

我们提出 BITSHARE，这是一种保留分组量化的整数网格和反量化器、但在每个组内采用不同精度的格式（图1）。它在比其速率细一位的网格上运行：以 3 比特表示 INT4，以 2 比特表示 INT3；并根据 GPTQ 已计算的海森矩阵对角线，将每个组的输入通道划分到三个固定位置的分箱中。高位、中位和低位分箱分别存储每个编码的 n、$n-1$ 和 $n-2$ 比特。每个组行存储一个标志位，为中位区间和低位区间中的每个编码提供最低位；每对低位区间权重存储一个比特，为其提供第二位。这种布线对每个层和模型都相同，而感知共享的 GPTQ 会选择共享值。解码器无需逐权重元数据或码本。它将比特路由到
编码，并运行不变的 INT-n 反量化器。BITSHARE-U 为每组采用相同的布局，因此布线是固定布线。BITSHARE-N 在每层的比特预算内为每组选择一种布局，因此每组需要一个 6 位索引，且解码器更大。两者分别存储每权重 3.164 和 2.156 比特，比 INT3 和 INT2 多 0.016 比特。

a) 贡献：我们提出了适用于大语言模型权重的静态比特共享：敏感度较低的权重，其低位通过布线读取存储的比特，而这些布线对每一层和每个模型都是固定的；我们还提出了敏感度感知分箱，可在每个组内调整精度而无需逐权重元数据（第三节B和第三节C）。据我们所知，这是首个面向大语言模型、共享结构预先固定且共享值由



符号说明。 n：完整编码的比特数（此处为 $n = 3$）。 s， z：组行的缩放因子和零点。 b_0， b_1：编码的两个最低位。 f：标志位，每个组行存储一个比特，用于指示每个中位区间和低位区间编码的 b_0。 c：成对比特，每两个低分箱权重存储一个比特，用于指示它们的 b_1。 U， D：高位区间和低位区间的大小。 $\delta = U - D/2$：布局的偏移量，以每组行比特数表示。

图 1. 大语言模型仅权重量化概览：在每个权重 2 比特、每层一行包含四组（每组 128 个位置）的情况下，不同格式如何存储该行。条形高度表示每个权重的编码位数。(a)-(c) 将每个编码位与其权重一同存储。在 (d)-(e) 中，BITSHARE 使用由 $n = 3$ 比特构成的网格：高位、中位和低位区间分别存储每个编码的 n、 $n-1$ 和 $n-2$ 比特，带阴影线的比特则由多个权重共享（图 3）。BITSHARE-U 在每组中使用 U8 D16；BITSHARE-N 则按组选择布局，并以每层 $\sum\delta \le 0$ 为依据。

编码器选择的整数权重格式。我们定义了两种模式：采用固定布线的 BITSHARE-U，以及通过动态规划精确分配组内布局的 BITSHARE-N（第三节D）；此外，我们还提出一种感知共享的 GPTQ，它能选择每个存储值（包括共享比特），无需微调或码本（第三节F）。我们在八种大语言模型上评估了这两种模式，并在存储量相同、困惑度相同以及直至崩溃等条件下与 SliM-LLM、GPTAQ 和 GPTQ 进行比较；同时还对该格式的各个部分进行了消融实验（第四节），并合成了解码器（第五节）。在 2 比特下，与 SliM-LLM 相比，BITSHARE-N 可将困惑度降低 19–70%；与 GPTAQ 相比，可降低 12.5–35.3%。在 SliM-LLM 的困惑度下，BITSHARE-U 每权重存储的比特数减少 13.6–19.4%。图 2 比较了在 WikiText-2 上，SliM-LLM、GPTAQ、GPTQ +裁剪以及 BITSHARE 的两种模式在 3 比特和 2 比特下相对于 FP16 的困惑度，展示了我们提出方法的有效性。

II. 背景

A. 大语言模型仅权重量化

a) 固定网格上的编码器：分组整数格式为每 128 个输入通道组成的组存储一个缩放因子和一个零点，并为每个权重存储一个编码（第 III-A 节），因此组内所有权重共用同一精度。GPTQ [2], 源自 OBS [15]和 OBQ [16], , 每次对一列进行舍入，并通过逆海森矩阵补偿其余列中的误差。GPTAQ [3]还会使每一层的输出与全精度模型的输出相匹配；OmniQuant [4]和 AffineQuant [5] 通过梯度下降学习量化器参数，而 QuaRot [17]通过旋转消除离群值。它们都保留 INT-n存储格式。

b) 码本与网格格式：QuIP# [10],、AQLM [11] 和 QTIP [12]通过格码本、加性码本或网格码，将 2 比特下的困惑度降得更低；这些方法通常结合哈达玛变换和微调，并通过查表或网格状态进行解码。在 QTIP 中，相邻权重通过重叠的网格窗口共享存储位。

B. 显著性与混合精度

权重 w_{ij} 的误差会经过输入通道 j 的激活能量缩放后传递至输出，该激活能量对应于$\mathbf{H} = 2\mathbf{X}\mathbf{X}^\top$ 的对角线。AWQ [1] 发现，根据激活幅值选取 1% 的通道并保留 FP16 精度，可将 OPT-6.7B 的就近舍入 INT3 困惑度从 23.54 降至 11.39；但 AWQ 改为重新缩放显著通道，以保持统一精度。OWQ [6], 、SpQR [7], 、SqueezeLLM [18] 和 PB-LLM [8] 以存储位置索引并采用稀疏解码路径为代价，将离群值保留为更高精度。HAWQ [19] 按层分配位宽。SliM-LLM [9] 按组分配 $N-1$、 N 或 $N+1$ 比特，为每组存储 2 位位宽；据报告，在 2 比特下，其解码吞吐量比 GPTQ 低 20–27%（第 IV-G 节）。每个组内的精度均匀一致。

C. 位级技术与共享比特

位串行加速器（如 Stripes [20],Pragmatic [21],Bitlet [22] 和 BitL [23]）会跳过或重排零比特，以加快对存储值的算术运算。BBS [13] 还会将一个权重组的低位比特裁剪为共享常量。Yang等人 [14]会重排 RRAM交叉开关行，使位列对变得相同，并为每对存储一列。这种方法无损，但取决于权重值，且需要

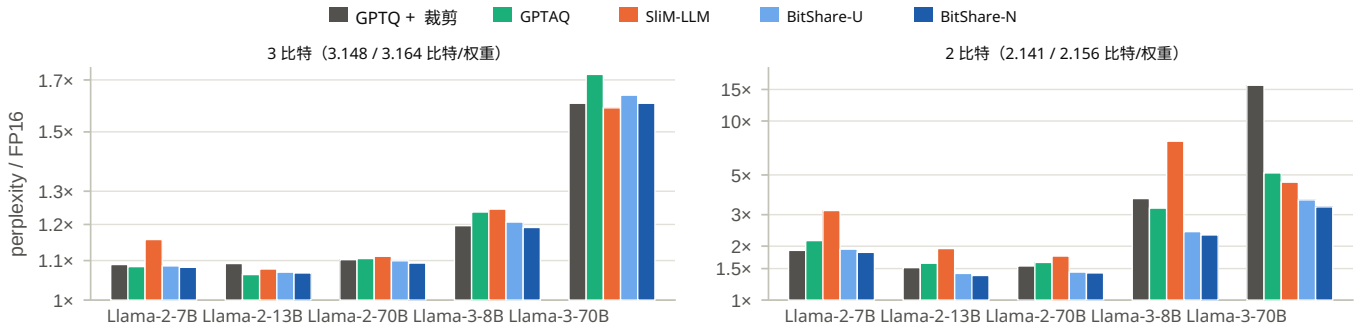


图 2. 3 比特和 2 比特下相对于 FP16 的 WikiText-2 困惑度（对数缩放）：SliM-LLM 和 GPTAQ、GPTQ + 裁剪，以及 BITSHARE 的两种模式（Llama-3-8B 使用 4096-token 窗口，其他模型使用 2048-token 窗口）。

存储索引。Shannonic [24]对量化张量进行熵编码，但无法将编码长度降至熵以下；微缩放格式 [25] 则在每个模块中共享一个指数，就像分组格式中已有的缩放因子一样。在机器学习之外，采用编码器选定值的固定共享结构十分常见。BC7纹理压缩 [26] 将一个 P 位共享比特存为多个颜色分量的最低有效位；Illinois Scan [27]和 EDT [28] 将少量测试比特扩展到许多扫描单元；总线反转编码 [29]为每组总线线缆设置一个反转比特，而 HIPR [30] 则在相邻查找表之间共享配置比特。两个值共享一个比特，是网格编码量化 [31]最简单的情形。

表 I 大语言模型仅权重量化格式比较。

方法	Codebook-free?	Index-free?	Single dequantizer?	Sensitivity-aware binning?	Intra-group precision?	Per-group layout?	Shared bits?
GPTQ、GPTAQ、AWQ	√	√	√				
OmniQuant、AffineQuant	√	√	√				
OWQ、SpQR、PB-LLM	√		√	√			
SliM-LLM	√	√	√	√			
QuIP#、AQLM	√	√					
QTIP	√	√		√			
BITSHARE-U（我们的方法）	√	√	√	√	√	√	
BITSHARE-N（我们的方法）	√	√	√	√	√	√	√

D. BITSHARE 的动机

表 I 总结了已有的格式。整数格式的解码成本较低，但会使组内每个权重具有相同的精度。精度可变的格式要么需要存储位置并采用稀疏路径，要么需要逐组宽度映射，而后者仍会使每个组使用同一个网格。码本和网格格式放弃了整数反量化器，而位级技术只共享量化后恰好相等的比特。据我们所知，尚无面向大语言模型的整数权重格式能够预先固定共享结构，并让二阶编码器选择共享值。BITSHARE 填补了这一空白。它在固定位置对每个组内的精度进行调整，无需逐权重元数据，并通过由 GPTQ 选择的共享比特承担相应开销，同时保留 INT-{v1}反量化器。我们的范围是无需基于梯度的训练的训练后标量整数格式，这也是 SliM-LLM 对比的设置（第 IV-A 节）。

我们的范围是无需基于梯度的训练的训练后标量整数格式，这也是 SliM-LLM 对比的设置（第 IV-A 节）。

III. 方法

BITSHARE 将每个线性层存储在标准分组量化的整数网格上，但为一个组中的权重分配不同数量的存储位。敏感度感知分箱将每个组的输入通道划分为三个分箱。较不敏感分箱中的权重仅存储其编码的较少比特，而未存储的每个比特都会被共享：通过固定布线，将其绑定到一个由多个权重读取的已存储比特上。一个组的各分箱大小构成其布局。BITSHARE 有两种模式，区别仅在于各组如何获得布局：BITSHARE-U（均匀）为每一层的每个组提供相同的布局，而 BITSHARE-N（非均匀）则从固定菜单中为每个组分配一个布局，使得需要更多比特的组可以从需要更少比特的组那里获取这些比特，同时总量保持不变。在两种模式下，GPTQ 的共享感知版本都会选择每个存储值。

A. 分组整数量化

线性层计算 $\mathbf{y} = \mathbf{W}\mathbf{x}$，其中$\mathbf{W} \in \mathbb{R}^{R\times C}$。分组量化将每一行划分为由 $G = 128$ 个连续输入通道组成的组，并为每个组行 (i, g)分配一个缩放因子 s_{ig}和一个零点 $z_{ig} \in \{0, \ldots, 2^n - 1\}$，在$n$比特网格上进行量化。权重 w_{ij} 存储为 n比特编码 q_{ij}，并重构为

$$\hat{w}_{ij} = s_{ig}\left(q_{ij} - z_{ig}\right). \tag{1}$$

采用非对称最小值-最大值约定和裁剪因子 $p \le 1$，$s_{ig} = p$(最大值$_j\, w_{ij}$ − 最小值$_j\, w_{ij}$)/$(2^n - 1)$，以及z_{ig} = 舍入$(-p$最小值$_j\, w_{ij}/s_{ig})$，并将结果限制在网格上，其中极值取自该组。INT-n 每个组行存储 nG 个编码位、一个 16 比特缩放因子和一个 n 比特零点，$n+(16+n)$每权重比特数：在 $G = 128$ 下，INT3 为 3.148，INT2 为 2.141。

BITSHARE 保持公式 1 的重构方式不变，并使用比其速率细一比特的网格：3 比特格式使用 INT4 网格（$n = 4$），2 比特格式使用 INT3 网格

（$n=3$）。将编码表示为其比特 $q=\sum_{k=0}^{n-1} b_k 2^k$ 后，每个权重存储其编码的高位比特，并从 sharedbits 中获取部分低位比特。

B. 敏感度感知分箱

权重 w_{ij} 的误差会传递到层输出，并受输入通道 j的激活值缩放，因此同一组中的各通道影响并不相同。我们通过层输出误差的海森矩阵对角线来衡量通道j 的敏感度：$h_j=[\mathbf{H}]_{jj}$，其中$\mathbf{H}=\frac{2}{N}\mathbf{X}\mathbf{X}^\top$ 是在 N 个校准词元上计算得到的矩阵，也是 GPTQ 已经计算出的矩阵 [2]。

组 g 具有如下布局（U_g，D_g）：其中 U_g 最敏感的通道构成高位区间 $\mathcal{H}_g$，D_g 最不敏感的通道构成低位区间 $\mathcal{L}_g$，其余 $M_g=G-U_g-D_g$通道构成中位区 $\mathcal{M}_g$。各区间占据组内的固定位置：位置 0、…、U_g-1 存放高位区间，U_g、…、$G-D_g-1$存放中位区，$G-D_g$、…、$G-1$存放低位区间；每个区间内的通道均保持原有顺序。因此，某个位置所属的区间，以及该位置上的哪些比特被存储、哪些比特被共享，仅取决于该位置和两个数值 U_g 与 D_g。分箱是在每个组内对输入通道进行置换 π，每层执行一次，并由所有 R 行共享。

置换在线下应用于$\mathbf{W}$ 的列。推理时，激活值通过该置换读取；或者等价地，解码后的行通过其逆置换写入。每个输入通道、每层需要一个 7 位索引，并由所有行共享。在 BITSHARE-N 中，每个线性层都有自己的置换，因为各层使用各自的布局；而在 BITSHARE-U 中，读取同一输入的层共享一个置换。按每个线性层计一个置换，对于 Llama-2-7B，置换会增加每权重 0.0012 比特。

C. 静态位共享

高位、中位区间和低位区间分别存储每个编码的 n、$n-1$ 和 $n-2$ 个比特，且始终是最高有效位。缺失的低位比特来自两种共享比特（图 3）：

- 成对比特：低位区间中的位置两两配对为（$G-D_g+2k$，$G-D_g+2k+1$）、$0\le k<D_g/2$，且每对中的两个权重都从一个存储比特读取其比特 b_1；
- 每个组行（i，g）中位区和低位区间权重的标志位 f_{ig}：比特 b_0 均为一个存储比特。

因此，组行（i，g）中位置 t 处权重的编码为

$$q=\begin{cases} v & t<U_g,\\ 2v+f_{ig} & U_g\le t<G-D_g,\\ 4v+2c_{igk(t)}+f_{ig} & t\ge G-D_g, \end{cases} \tag{2}$$

其中，v是权重的存储字段，包含 n、$n-1$ 或 $n-2$ 个比特，而 $k(t)=\lfloor (t-G+D_g)/2\rfloor$ 是其权重对。

共享约束的是编码，而不是丢弃比特。中位区间的权重可以采用任何奇偶性为 f_{ig} 的编码：这是一个步长为细步长两倍的$(n-1)$ 比特网格。低位区间的权重可以采用任何模 4 同余于 $f_{ig}+2c_{igk}$ 的编码：这是一个步长为细步长四倍的网格，其偏移量按权重对选定。所有共享比特都可编程。对于每一行，编码器都会将每个成对比特设为一个值，使权重对的误差按照

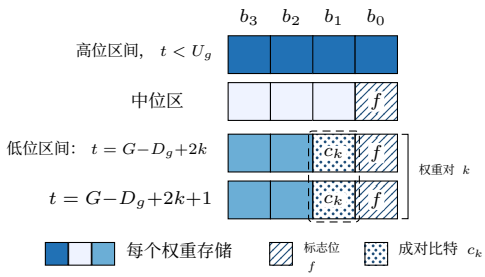


图 3. 3 比特格式：组 g中每个区间内一个权重的 INT4 码 $q=b_3b_2b_1b_0$ 的比特。实心比特与权重一同存储，颜色与其所在区间一致。带图案的比特从共享的存储比特中读取：每个 f 都是组行的一个标志位 f_{ig}，而低位分箱对中的两个 c_k 共用一个成对比特 c_{igk}。组行还存储其缩放因子 s_{ig} 和零点 z_{ig}。2 比特格式会舍弃每个区间中的 b_3（INT3 网格）。

层输出中的权重最小化，因此当两个权重需要不同的值时，该比特会分配给误差影响更大的那个权重。编码器还会设置每个组行的标志位。相比之下，从编码中丢弃的比特，对于每个缺少该比特的权重而言都是相同的。

该标志位使编码器能够为每个组行选择组取值范围的哪一端不被中位区间和低位区间所用的较窄网格覆盖（每个组行对应一位），而不是将其固定为零点的奇偶性（附录 B）。

采用布局 $(U,\ D)$ 的组行包含 $nU+(n-1)M+(n-2)D+D/2$ 编码和成对比特，即

$$(n-1)G+\delta(U,D),\qquad \delta(U,D)=U-\tfrac{D}{2}, \tag{3}$$

因此，每比特高位区间的额外比特由低位区间在两个位置上分摊。参考布局 $U=8$, $D=16(M=104)$ 含有 $\delta=0$：恰好是 INT-($n-1$) 的有效载荷。BITSHARE-U 在每个组中都使用它。采用 $\delta<0$ 的布局所需比特更少。在 $B>2$ 个低位区间位置之间共享 b_1，或将其舍弃，可以进一步降低速率（附录 B）。两种模式都使用权重对 ($B=2$)；IV-D 和 IV-E 节仅使用其他变体来扩展 BITSHARE-U 的率失真曲线。

D. 每组布局（BitShare-N）

BITSHARE-U 的单一布局适合平均组，但各组的敏感度在通道间的分布、总体水平和范围各不相同，因此相同的比特数在某些组中比在其他组中能换来更高的精度。因此，BITSHARE-N 允许每个组从一个包含 44 种布局的菜单 Λ 中选择布局，该菜单对所有层和模型都相同。菜单包含 $U\in\{0,4,8,12,16,24,32\}$和 $D\in\{0,16,32,48,64,96\}$ 的所有组合，以及采用δ的布局（$U,\ D$）$=0$（$U=D/2$，从 U0 D0 到 U32 D64）；这些布局的低位区间都使用权重对。它们的偏移量 δ 范围为每组行 -48 至 $+32$比特（图1）。BITSHARE-U 是菜单仅含一种布局的特殊情况，即 $\Lambda=\{$U8 D16$\}$，除非需要更低的速率。

对于每个组 g 和布局 $\ell\in\Lambda$，编码器都会测量选择误差

$$E_g(\ell)=\sum_{i=1}^{R}\sum_{j\in g} h_j\left(w_{ij}-\hat{w}_{ij}^{(\ell)}\right)^2, \tag{4}$$

其中，$\hat{}w^{(\ell)}$ 会根据布局 ℓ，将每个权重舍入到其所在区间允许的最近编码；缩放因子、零点和标志位则由

表II 每组行比特数（$G = 128$，权重对），适用于布局（U，D）的组，以及参考布局 U8 D16（$M = 104$）；BITSHARE-U 在每个组中都使用该参考布局。另列出每权重比特数。在 BITSHARE-N 中，一层的偏移量 δ 之和至多为 0，因此其总量至多与参考布局相同。最后一行给出了组大小、缩放因子和零点相同时 INT3 和 INT2 的每权重比特数。

字段	布局（U，D）	U8 D16	
		$n = 4$	$n = 3$
高箱编码	nU	32	24
中箱字段	$(n-1)M$	312	208
低箱字段	$(n-2)D$	32	16
成对比特	$D/2$	8	8
缩放因子（FP16）	16	16	16
零点	n	4	3
Flag	1	1	1
总比特数	$(n-1)\ G+\delta+n+17$	405	276
每权重比特数		3.164	2.156
INT3 / INT2		3.148	2.141

同一范围搜索确定每个组行的值，该搜索也用于设置组参数（第三节F）；每个权重对的成对比特则取误差较小的值。随后，层的布局求解

$$\min_{\ell_1,\ldots,\ell_{C/G}\in\Lambda} \sum_g E_g(\ell_g) \quad \text{subject to} \quad \sum_g \delta(\ell_g) \le 0, \quad (5)$$

因此，该层存储的数据量至多为参考负载，即每权重（$n-1$）比特；只有在能降低该层误差时，一个组才能从另一个组借用比特。公式 5 是一个针对该层各组的多重选择背包问题。由于所有偏移量都是 4 的倍数，因此可以对偏移量的累计和进行动态规划，以精确求解该问题，且状态数不超过 $20\,C/G+1$。每层都根据自身的敏感度求解公式 5，因此各层之间以及同一层的各组之间，布局也各不相同。布局以每组一个 6 位索引的形式存储，该索引由该层的所有 R 行共享，因此对于 $R \ge 4096$，每权重占用 $6/(GR) \approx 10^{-5}$ 比特。BITSHARE-U 不存储索引。

E. 存储格式与解码器

表II列出了组行的字段。在采用 U8 D16 的 BITSHARE-U 中，以及当某一层的偏移量之和为零时的 BITSHARE-N 中，该层的编码和成对比特与 INT-$(n-1)$ 相同。在此基础上，该格式还存储标志位和一个额外的零点比特，因为 z_{ig} 是更细网格的编码。与 INT-$(n-1)$ 相比，每权重增加 $2/128 = 0.016$ 比特，因此两种模式下每权重的总比特数分别为 3.164 和 2.156。BITSHARE-N 的布局索引还会增加约 10^{-5}。

组行根据自身字段和所属组的布局 $(U_g,\ D_g)$进行解码。在 BITSHARE-U 中，所有组的布局都相同；在 BITSHARE-N 中，则根据组在菜单中的 6 位索引查表得出。菜单是一个由所有层和模型共享、含有 44 个条目的表。解码器读取 s_{ig}, z_{ig} 和 f_{ig}，然后读取 U_g n-比特、M_g $(n-1)$-比特和 D_g $(n-2)$-比特字段，以及 $D_g/2$ 成对比特，按照公式 2 构成编码，并按照公式 1 对其反量化，与 INT-n完全相同。构成编码不需要进行算术运算：只需将某个位置与所属组的边界进行两次比较，即可确定它所在的区间。在 BITSHARE-U 中，所有组的边界都相同，因此布线是固定的。在 BITSHARE-N 中，不同布局的组行大小各异，但同一层的每一行都具有相同的布局序列，因此组在行内的偏移量是该层布局索引的前缀和，每层只需计算一次。该格式不包含任何逐权重或逐模块元数据，也不包含权重值码本或查找表。

F. 感知共享的 *GPTQ*

编码器会选择每个存储值。与 GPTQ [2], 类似，它按照顺序使用校准数据量化 Transformer 块；在每个块内，则按执行顺序量化线性层，每层使用此前已量化层所产生的输入。对于输入为$\mathbf{X}$的层，GPTQ 按列最小化 $\|(\mathbf{W} - \hat{\mathbf{W}})\mathbf{X}\|_F^2$：一列量化后，通过海森矩阵逆在尚未量化的列中补偿其误差。BITSHARE-N 首先选择该层的布局，而 BITSHARE-U 为每个组指定一种布局。随后，两种模式都会修改 GPTQ 的三个步骤，如算法 1（附录 A）所示。

a) 循环中采用该格式的组参数：在列循环开始前，固定每个组行的缩放因子、零点和标志位，与采用静态分组的 GPTQ 相同。对于每个裁剪因子 $p \in \{1.00, 0.98, \ldots, 0.80\}$ 和每个标志位取值 $f \in \{0, 1\}$，将组行中的每个权重舍入到其区间允许的最近编码；每个低位分箱对则选取误差较小的成对比特。保留灵敏度加权误差最小的候选项 $\sum_j h_j(w_{ij} - \hat{w}_{ij})^2$。因此，标志位与范围共同选定，使 11 点范围搜索的搜索量加倍。布局的选择误差（公式 4）就是此次搜索的误差在各行上的总和。

b) 按敏感度排序的量化单元：GPTQ 对量化单元进行量化：高位和中位区间中的单列，以及低位区间中的双列（对于由 B 共享一个比特的 B 列模块，单列表示丢弃一个比特）。量化单元按照整个层中的敏感度从高到低依次量化（act-order）；权重对或模块按其中敏感度最高的列排序。因此，最敏感的通道会最先量化，同时保留尽可能多的列来补偿其误差。

c) 根据 *GPTQ* 目标函数选择共享比特：对于单列，按行逐一舍入到标志位所允许的区间内最接近的编码。对于权重对，每行都会尝试成对比特的两种取值 c。第一列会在满足 $b_1 = c$ 的条件下舍入到最接近的编码，并将其误差传播到第二列，就像 GPTQ 将误差传播到后续任何一列一样，然后再对第二列进行舍入。保留目标函数值较小的取值 $\sum_{j\in\text{单元}}(w_{ij} - \hat{w}_{ij})^2/d_j^2$，其中 d_j 是 GPTQ 用于加权误差的 $\mathbf{H}^{-1}$ 的乔列斯基分解因子的对角线。随后将该单元的误差传播到其余列。

IV. 结果

A. 评估设置

模型。Llama-2（7B、13B 和 70B） [32]，Llama-3（8B 和 70B） [33]以及通义千问2.5（0.5B、1.5B 和 3B） [34]。Transformer 块中的所有线性层均经过量化。嵌入层和输出层保持 FP16。

表 III 在 SliM-LLM 的协议下（在 WikiText-2 上进行校准）的 WikiText-2 困惑度，分别采用 3 比特和 2 比特。Llama-3-8B 使用 4096-token 窗口进行评估，与 SliM-LLM 的论文一致；其他模型则使用 2048令牌窗口。最低困惑度值以**粗体**显示。

	3 比特						2 比特					
		Llama-2			Llama-3			Llama-2			Llama-3	
方法	b/w	7B	13B	70B	8B	70B	b/w	7B	13B	70B	8B	70B
FP16	16	5.47	4.88	3.32	5.75	2.85	16	5.47	4.88	3.32	5.75	2.85
RTN	3.148	6.66	5.51	3.97	27.91†	11.84	2.141	4.2e3	122.1	27.27	1.9e3†	4.6e5
AWQ	3.148	6.24	5.32	–	8.22†	4.81	2.141	2.2e5	1.2e5	–	1.7e6†	1.7e6
QuIP	–	–	–	–	–	–	–	39.73	13.48	6.64	84.97†	13.03
PB-LLM	–	–	–	–	–	–	–	25.37	49.81	NaN	44.12†	11.68
OmniQuant	3.148	6.03	5.28	3.78	–	–	2.141	11.06	8.26	6.55	–	–
AffineQuant	3.148	6.08	5.28	–	–	–	2.141	10.87	7.64	–	–	–
SliM-LLM+	3.148	5.94	**5.11**	**3.35**	–	–	2.141	10.87	7.59	6.44	–	–
SliM-LLM ★	3.148	6.33	5.26	3.69	7.16	**4.53**	2.141	17.31	9.47	5.85	44.35	12.98
GPTAQ ★	3.148	5.93	5.19	3.67	7.11	4.91	2.141	11.76	7.84	5.39	18.76	14.62
GPTQ + 裁剪 ★	3.148	5.96	5.33	3.66	6.88	4.58	2.141	10.37	7.41	5.15	21.22	45.13
BITSHARE-U	3.164	5.94	5.22	3.65	6.94	4.67	2.156	10.53	6.88	4.76	13.87	10.34
BITSHARE-N	3.164	**5.92**	5.21	3.63	**6.85**	4.58	2.156	**10.11**	**6.70**	**4.71**	**13.29**	**9.45**

† SliM-LLM 报告的 Llama-3-8B 基线与 2048令牌窗口相匹配。每权重比特数：每个权重的比特数。–：未报告。★ 我们使用 SliM-LLM 的已发布代码进行运行 [36], GPTAQ [37] 和 GPTQ [35]。

方法。 每种方法都会将 $G = 128$ 输入通道量化为组，并为每个组行使用一个 FP16缩放因子和一个整数零点。

- *GPTQ* [2]: 使用最小最大范围、带静态分组的 act-order、阻尼 $0.01h$ 和顺序层的 INT3 或 INT2（算法 1）。在测试层上，我们的实现生成了与已发布 GPTQ [35]相同的编码。

- *GPTQ* + 裁剪：采用 BITSHARE 范围搜索的 GPTQ，$p \in \{1.00, 0.98, \ldots, 0.80\}$；在 2 比特下也使用 $G = 64$，其每权重存储的比特数比 BITSHARE 多 0.125。
- *SliM-LLM* [9]: 根据显著性，每组使用 2、3 或 4 比特，每层平均使用 3 比特（在 1–3 比特设置下平均为 2 比特；在 2 比特设置下亦然），并采用其自有的 GPTQ。我们在 Llama-2-7B/13B 和 Llama-3-8B 上运行其已发布代码 [36]，并使用已发布的组宽度；在 70B 模型上则使用其自有的宽度搜索，因为这些模型均未发布（附录 C-A）。我们报告了自己的运行结果。

- *GPTAQ* [3] 在 GPTQ 的更新中加入一项，使每层的输出与全精度模型的输出相匹配，从而补偿前面各层的误差。我们使用 act-order，在 INT3 或 INT2 网格上运行其已发布代码 [37]，并采用其自有的范围搜索（附录 C-A）。

- 仅报告结果：按照 SliM-LLM 的协议，我们还列出 SliM-LLM 报告的 RTN、AWQ [1],QuIP [38]和 PB-LLM [8], 的困惑度，这些方法无需梯度训练；此外，还列出 OmniQuant [4], AffineQuant [5] 和 SliM-LLM+的困惑度，这些方法通过梯度下降学习其量化器参数。

- BITSHARE：INT4 网格上的 3 比特格式和 INT3 网格上的 2 比特格式，包含两种模式：BITSHARE-U，除非另有说明，否则每组均采用 U8D16；以及 BITSHARE-N，其每组布局由公式 5 确定。

每权重比特数统计组行中的每个字段：INT3、3 比特 SliM-LLM 和 GPTAQ 为 3.148，INT2、2 比特 SliM-LLM 和 GPTAQ 为 2.141。BITSHARE 的两种模式分别存储 3.164 和 2.156（表II）。置换以及 BITSHARE-N 中的布局索引，在 Llama 模型上每权重增加 0.0006-0.0013 比特，在通义千问模型上增加 0.002-0.005 比特。GPTAQ 在按 act-order 形成组时为每列存储的组索引约增加 0.001 比特。这些比特均未计入。

校准与评估。 与 SliM-LLM 和 GPTAQ 的比较，以及所有 70B 模型的运行，均遵循 SliM-LLM 的协议：使用 GPTQ 的加载器和 SliM-LLM 的随机种子，从 WikiText-2 训练集划分中抽取 128 个窗口，每个窗口包含 2048 个词元。对于不同模型系列，我们还按照 AWQ [1]的方法，在 Pile [39], 验证集划分中使用 128 个序列进行校准，每个序列包含 2048 个词元。我们报告 WikiText-2 测试集划分上的困惑度（按每个包含 2048 个词元的窗口计算） [40]，以及 GPTQ 的 C4 验证文本前 256 个窗口上的困惑度（每个窗口包含 2048 个词元） [41]。附录 C 补充了按每个窗口包含 4096 个词元计算的 WikiText-2 结果。所有运行均使用 NVIDIA RTX A6000 图形处理器。

B. SliM-LLM 和 GPTAQ

表 III 将我们的方法与 SliM-LLM 和 GPTAQ 在 SliM-LLM 的协议下进行比较，其中包括 70B 模型。图 2 和表 VIII（附录 C）展示了这些实验相对于 FP16 的结果，以及在三个测试集上的表现。在 2 比特下，BITSHARE-N 在我们测试的全部五个模型上都优于其他所有方法：比 SliM-LLM 低 19-70%，比 GPTAQ 低 12.5-35.3%，比 GPTQ+ 裁剪低 2.5-79.1%。BITSHARE-U 在全部五个模型上也优于 SliM-LLM 和 GPTAQ。两种模式的困惑度都低于 SliM-LLM 报告的所有方法，包括三种通过梯度下降学习量化器参数的方法：在 Llama-2-13B 和 Llama-2-70B 上，这些方法的最佳结果分别为 7.59 和 6.44，而 BITSHARE-N 为 6.70 和 4.71，BITSHARE-U 为 6.88 和 4.76。在 3 比特下，BITSHARE-N 在 Llama-2 7B、13B 和 70B 以及 Llama-3-8B 上比 SliM-LLM 低 1.0-6.4%，在 Llama-3-70B 上则高 1.1%；在该模型上，它与 GPTQ+ 裁剪的 4.58 持平。在 SliM-LLM 报告的方法中，只有 SliM-LLM+ 在 3 比特下优于 BITSHARE-N，涉及 Llama-2-13B 和

表 IV：相同存储条件下的困惑度（↓），在 Pile 上校准。B/W：每个权重的比特数。最低困惑度值以**粗体**显示。

		WikiText-2						C4					
		Llama-2		Llama-3	Qwen2.5			Llama-2		Llama-3	Qwen2.5		
方法	b/w	7B	13B	8B	0.5B	1.5B	3B	7B	13B	8B	0.5B	1.5B	3B
FP16	16	5.47	4.88	6.14	13.07	9.27	8.03	7.26	6.73	9.44	20.38	15.12	13.36
3 比特													
GPTQ	3.148	6.26	5.41	7.96	19.70	12.13	10.01	8.38	7.41	12.82	29.21	18.67	15.78
GPTQ + 裁剪	3.148	6.06	5.45	7.82	19.11	11.23	9.45	8.07	7.52	12.09	28.51	17.71	15.15
BITSHARE-U	3.164	**6.01**	5.34	7.43	17.60	**10.91**	9.29	**8.02**	7.30	11.72	26.43	17.39	14.94
BITSHARE-N	3.164	6.02	**5.29**	**7.41**	**17.25**	10.94	**9.26**	**8.02**	**7.28**	**11.68**	**26.11**	**17.32**	**14.91**
2 比特													
GPTQ + 裁剪	2.141	10.49	8.31	39.51	129.9	32.87	21.74	13.46	11.10	75.00	158.5	47.32	29.75
GPTQ + 裁剪， G=64	2.281	9.32	7.66	33.80	**56.43**	24.81	17.20	**12.02**	10.30	68.22	78.43	35.07	24.33
BITSHARE-U	2.156	9.25	7.43	16.35	63.63	20.61	15.44	12.06	10.26	22.86	83.66	31.74	22.96
BITSHARE-N	2.156	**8.95**	**7.31**	**15.19**	58.02	**20.60**	**15.29**	12.07	**10.11**	**21.83**	**77.45**	**31.14**	**22.58**

Llama-2-70B（分别为 5.11 和 3.35，而 BITSHARE-N 为 5.21 和 3.63）。在 Llama-2-7B 上，GPTAQ 与 BITSHARE-N 持平（5.93 对 5.92）；在 Llama-2-13B 上则优于它（ 5.19 对 5.21 ）。在另外三个模型上，BITSHARE-N 低 1.0-6.8%。在 10 个单元中，GPTQ + 裁剪本身有 7 个单元的表现优于 SliM-LLM。

在表 III 的全部 10 个单元中，BITSHARE-N 的结果都优于 BITSHARE-U。在 Llama-3-70B 的 2 比特设置下，其组内布局将困惑度从 10.34 降至 9.45。在 2 比特设置下，运行结果还取决于浮点舍入：在 Llama-2-7B 和 Llama-3-8B 上重复运行（附录 C-B）时，结果波动幅度与两种模式之间的差异一样大；不过，每次运行中 BITSHARE-N 都优于 GPTQ + clip，且两种模式的结果都远优于 SliM-LLM。

C. 在 *Pile* 上校准

表 IV 比较了在 Pile 上校准时 BITSHARE 的两种模式与 GPTQ 在相同存储开销下的表现，测试模型为 Llama-2、Llama-3 和通义千问 Qwen2.5。在所有模型、两种位宽和两个测试集上，BITSHARE 的两种模式都优于 GPTQ + clip。在 3 比特设置下，BITSHARE-N 在 WikiText-2 上的困惑度比它低 0.7–9.7%，在 C4 上低 0.7–8.4%。增益在通义千问 Qwen2.5-0.5B 和 Llama-3-8B 上最大，这两个模型的困惑度受 GPTQ + clip 恶化最严重。在 2 比特设置下，GPTQ + clip 会使 Llama-2 上 FP16 的困惑度增加 70–92%，并使 Llama-3-8B 和通义千问 Qwen2.5-0.5B 上的困惑度分别增至原来的 6.4 倍和 9.9 倍；相比之下，BITSHARE-N 在 WikiText-2 上的困惑度比它低 12–62%，在 C4 上低 9–71%。采用 64 个一组的 GPTQ +clip 会多存储 0.125 每权重比特数，但在 12 个单元中的 10 个里，其表现仍不如 BITSHARE-N。

在每个单元格中，相较于 GPTQ，+ 裁剪带来的大部分收益已由 BITSHARE-U 实现。BITSHARE-N 的组内布局在此基础上进一步提升了 2 比特下的效果：在 WikiText-2 上，BITSHARE-N 在所有模型上的困惑度都低于 BITSHARE-U，平均低 3.6%，在通义千问 Qwen2.5-0.5B 上最多低 8.8%。在 C4 上，六个模型中有五个的困惑度更低，平均低 2.8%；在 Llama-2-7B 上则基本持平（12.07 对 12.06）。在 3 比特下，平均变化为 -0.5%。

D. 相同困惑度

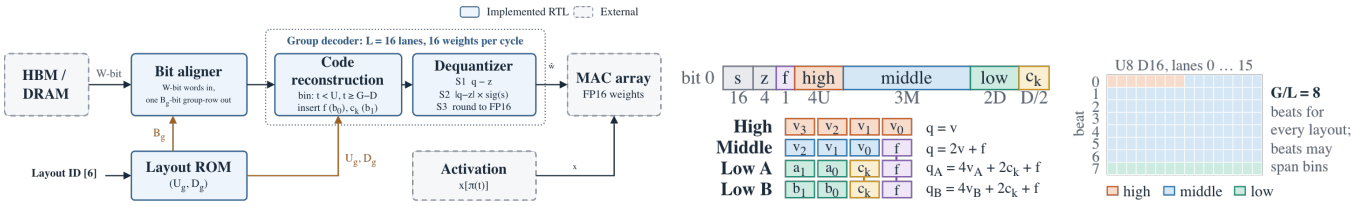
布局以 $\delta < 0$ 比特换取困惑度。在 SliM- LLM 的协议下，我们评估 BITSHARE-U：在 INT4 网格中，每个组采用一个布局，U8 下为 $D \in \{16, 48, 96, 120\}$，在 INT3 网格中为 $D \in \{16, 48, 80, 120\}$，并且在$D = 120$ 下，每四个低分箱权重共享一个比特 b_1，而不是两个。总量分别从 3.164 降至 2.523，以及从 2.156 降至 1.516 每权重比特数。表 V 给出了这条曲线达到 SliM-LLM 的编码所对应困惑度时的存储量；该曲线按困惑度进行对数线性插值，附录 C-C 中的图 6 展示了这些曲线。在 2 比特下，BITSHARE- U 在 WikiText-2 上所需的每权重比特数减少了 17.9-19.4%，为 1.726-1.758，而不是 2.141；在 C4 上则减少了 21.6% 或更多。在 3 比特下，节省幅度为 2.1-14.7%。对于 70B 模型，在这些模型上 SliM- LLM 的编码会搜索其自身的位宽，2 比特下的节省幅度在 Llama-2-70B 上为 18.2%，在 Llama-3-70B 上为 13.6%；3 比特下，Llama-2-70B 上为 1.9%。在 Llama-3-70B 上，采用 U8 D16（3.164 每权重比特数）的 BITSHARE-U 高于 SliM-LLM 的 3 比特困惑度。使用 U8 D0 和 U32 D0 后，曲线继续上升（分别为 3.227 和 3.414），并在 3.200 处达到该困惑度，比 SliM-LLM 高 1.7%。这些节省幅度属于 BITSHARE-U。在 WikiText-2 上，BITSHARE-N 在 3.164 和 2.156 每权重比特数下仍然更低。

表 V 相同困惑度下节省的存储：在 SliM-LLM 的协议下，当 BITSHARE-U（每个组采用一种布局）的率失真曲线达到 SliM-LLM 编码的困惑度时，相较于 SliM-LLM（3.148 和 2.141），每权重比特数的减少量。对于 70B 模型，SliM-LLM 编码会搜索其自身的位宽，且仅评估 WikiText-2。 $\geq$：SliM-LLM 的困惑度高于此曲线；该曲线在每权重 1.516 比特处结束；负值表示比 SliM-LLM 使用更多比特。

	3 比特		2 比特	
模型	WikiText-2	C4	WikiText-2	C4
Llama-2-7B	14.7%	5.7%	19.4%	28.5%
Llama-2-13B	2.1%	3.2%	18.5%	21.6%
Llama-3-8B	5.2%	10.7%	17.9%	$\geq$29.2%
Llama-2-70B	1.9%	–	18.2%	–
Llama-3-70B	$-$1.7%	–	13.6%	–

E. 直至崩溃

我们在 SliM-LLM 的协议下，使用 U0 D128 将 BITSHARE-U 在 INT3 网格上的存储开销扩展到每个权重低于 1.516 比特，其中 b_1由 4/8/16 个权重共享或丢弃（1.406/1.281/1.219/1.156



(a) 层解码器。 (b) 代码重建。 (c) 单个组行。

图 4. BITSHARE 解码器硬件实现。(a) 每个组的布局索引会选择 (U_g, D_g) 和组行大小 B_g；位对齐器从打包层中提取每个组行，而组解码器则重建并反量化其编码。激活值通过 π 读取。(b) 代码重建： f 提供中位区间和低区间代码的 b_0， c_k 提供低位分箱对中两种代码的 b_1 。(c) 输出可能跨越多个区间，因此每种布局都将每个组行中 L 个权重的输出取为 G/L 。

比特/权重）；使用 U8 D120 进行八路共享或丢弃（1.398/1.281），以及使用包含 256 个权重的组的 U0 D256 并丢弃 b_1（1.078）。

附录 C-C 的表 X 给出了 WikiText-2 困惑度达到 FP16 的两倍和十倍时的速率，每种速率下采用这些格式中表现最佳的一种。在 Llama-2 中，较大的模型能维持更久：从 7B 到 70B，困惑度分别在每权重 2.02、1.72 和 1.61 比特时翻倍，在 1.36、1.22 和 1.17 时达到 FP16 的十倍。Llama-3-70B 在每权重 2.91 比特时困惑度就已翻倍，而 Llama-3-8B 为 2.30；但此后其性能下降得更慢，在 1.41 时达到 FP16 的十倍，而 Llama-3-8B 为 1.68。共享 b_1 比丢弃它能让模型更长时间保持可用：在每权重 1.281 比特时，Llama-2 7B、13B 和 70B 中，每八个权重共享 b_1 时的困惑度分别比不使用b_1 低 2.1、2.7 和 1.7 倍；Llama-3 8B 和 70B 中则分别低 32 倍和 1.2 倍。在达到 FP16 的三倍之前，每条曲线都遵循 $\ln d = a_m + \beta_m b$，其中 $d = \ln(\text{ppl}/\text{ppl}_{\text{FP16}})$。利用其他模型估算斜率，只需对某个模型运行一次，就能预测其余曲线，困惑度的中位误差为 5.2%（附录 C-C）。

F. 消融实验

附录 C-D 中的表 XI 从 GPTQ + 裁剪的 INT2 网格开始，逐一剥离 2 比特格式的各个组成部分。去掉分箱（U0 D0）后，在 Llama-2 上的困惑度比 BITSHARE-U 高出不到 0.6%，但在 Llama-3-8B 和通义千问模型上则高出 8–65%；在这些模型中，每组有少数几个通道需要使用高区间的完整 INT3 网格。将 b_0 比特从零点的奇偶性而非标志位中获取，可为每个权重节省 1/128 比特，但会使困惑度上升 5–30%。每组采用等比特的布局，平均而言可使困惑度比 BITSHARE-U 降低 2.6%，但并非对所有模型都如此；而像 BITSHARE-N 那样依据公式 5 在组之间移动比特，则能使所有模型的困惑度降低，平均降幅为 3.6%。附录还提供了详细信息，并展示了 BITSHARE-N 选择的布局（图 7 ）。

G. 编码与解码开销

a) 编码： BITSHARE-N 根据权重以及 GPTQ 无论如何都会累积的海森矩阵对角线来选择布局，并且两种模式都只需经过一次 GPTQ 过程即可找到编码。在一块 48GB 图形处理器上，使用 GPTQ +裁剪量化 Llama-2-70B 或 Llama-3-70B 需要 1.8–1.9 小时，使用 BITSHARE-U 需要 2.2–2.3 小时，使用 BITSHARE-N 需要 3.1 小时，在两种宽度下均如此。SliM-LLM 根据线性层在校准集上的输出 KL 散度，为每种候选分配方案评分。仅这一搜索过程，其编码每个解码器层就需要 32–38 分钟；在一块图形处理器上，每个模型和宽度还需要 43–51 小时。

b) 解码： 假设权重在批次大小为一时每个词元仅流式传输一次，则每权重比特数会限制受带宽限制的解码吞吐量。两种模式在组记录中均使用 3.164/2.156 每权重比特，约比 INT3/INT2 和 SliM-LLM 高 0.016。然而，在 WikiText-2 上达到 SliM-LLM 的 2 比特困惑度时，BITSHARE-U 读取的比特数减少了 13.6%–19.4%（表 V），附录 D-E 将这些节省转化为加速比上界。两种模式都没有逐权重元数据或码本。在 BITSHARE-U 中，每个组使用相同的布局，因此每个字段都位于固定偏移量处，无需布局表。BITSHARE-N 为每个组读取一个 6 位布局索引，并在依赖布局的偏移量处选择各字段。第五节在硬件中实现了其解码器。SliM-LLM 的组大小也各不相同；在三个模型和一个 A800 图形处理器上，使用其自有内核时，它报告的图形处理器吞吐量比 2 比特 GPTQ 低 20%–27%（ [9] 的表 4）。

V. 硬件实现

我们使用可综合的 SystemVerilog 编码实现了 BITSHARE 的解码器，对其进行了验证，并在相似约束下将其资源开销与 INT-n 和 SliM-LLM 进行了比较。图 4 展示了从压缩流到 FP16 权重的数据通路（图 4(a)）、利用存储比特和共享比特进行重建的过程（图 4(b)），以及每组行输出 G/L 个结果且与布局无关的输出调度（图 4(c)）。

A. 微体系结构

在解码器实现中，组行无填充地首尾相接存储；每组一个由层中所有行共享的 6 位布局索引，为每个组行指定其布局（U_g、 D_g）和大小 B_g。位对齐器从打包流中截取每个组行，组解码器根据公式 2 重建 n 位编码，并根据公式 1 将其反量化为 binary16，其过程与 INT-n 完全相同。对于每种布局，它每周期输出 L 个权重，每个组行耗时 G/L 个周期，因此对于 $G/L \geq 2$，通过 W 位内存接口，可持续达到 min(L、 WG/B) 个权重/周期，其中 B 是组行大小的平均值；在 $G = 128$ 和 $L = 16$ 下，64 位接口可跟上每种布局。附录 D-A 包含详细实现。

B. 硬件成本

1) 基线：我们比较三种具有相同接口、输入缓冲、三级流水线和 binary16 反量化器的解码器；它们在组大小为 128 时均可达到每周期 16 个权重：INT-n（GPTQ 存储），采用固定的 n 位字段和通道偏移；SliM-LLM，每权重使用 $n-1$、n 或 $n+1$ 比特，由每组共享的 2 位编码在各组间选择，采用密集打包的记录和仿射重建；以及采用组内布局和双路共享的 BITSHARE。SliM-LLM 和 BITSHARE 对可变大小的记录需要进行位对齐。我们还为 INT-n 配备了相同的对齐器，尽管其固定大小的记录允许采用更简单的设计。面积是映射后单元面积之和，不包括布局、布线或时钟树。逻辑延迟是 ABC 报告的关键组合路径。

2) 设置：我们使用开源 Yosys 和 ABC [42], [43] 工具，采用典型工艺角（25°C、1.80 V）下的 SkyWater 130 纳米（sky130 fd sc hd）[44] 标准单元库，对每个设计进行扁平化和综合。面积为映射后的标准单元面积之和，不包括布局、布线及时钟树开销。报告包括关键组合逻辑延迟，但不包括寄存器、设置和互连延迟。

表 VI 相同吞吐量下的解码器面积（16 个权重/周期，$G = 128$），SKYWATER 130 纳米。**解码器**：组解码器加上位对齐器，以及对于 BITSHARE 而言的布局只读存储器。触发器和逻辑延迟均针对组解码器计算。

Rate	格式	面积（μm²）		与 INT-n 对比	触发器	逻辑电路 延迟（ns）
		组解码器	解码器			
3 比特	INT3	100,693	183,286	1.00×	1,795	6.39
	SliM-LLM	129,728	237,218	1.29×	2,169	7.36
	BITSHARE	210,243	318,134	1.74×	2,327	7.88
2 比特	INT2	80,106	139,928	1.00×	1,425	6.15
	SliM-LLM	105,057	188,672	1.35×	1,799	6.13
	BITSHARE	175,384	258,311	1.85×	1,957	7.63

3) 结果：表 VI 展示了 BITSHARE 解码器实现的硬件开销。在相同吞吐量下，BITSHARE 解码器在 3 比特时的面积为 SliM-LLM 解码器的 1.34×，在 2 比特时为 1.37×；其面积为 INT-n 的 1.74×和 1.85×。其逻辑延迟为 INT-n 的 1.23× 和 1.24×，为 SliM-LLM 的 1.07×和 1.24×。

在 3 比特下，三个分组解码器的触发器面积分别为 53.8（INT3）、65.0（SliM-LLM）和 69.8kμm²（BITSHARE），而逻辑面积分别为 46.9、64.7 和 140.5 kμm²。INT-n 通道读取固定的位片，SliM-LLM 通道则在三个固定步长中进行选择。BITSHARE 通道在依赖布局的偏移量处选择其字段，且活动分组每拍前进由布局决定数量的比特。三者共享同一个 binary16 反量化器，而 SliM-LLM 和 BITSHARE 的位对齐器面积几乎相同。

这种开销支持不同的质量与存储权衡。在存储量相当的情况下，BITSHARE 增加约 0.016 每权重比特数，并在大多数评估设置下实现更低的困惑度。在困惑度相匹配的情况下，这些配置在大多数情况下可减少存储（表 V）和权重流量（附录 D-E）。解码电路可在不同组和层之间复用，因此其面积取决于实例化的解码引擎数量，而不是所存储权重的数量。因此，仅解码器面积比并不代表整机加速器开销：后者还取决于计算阵列、存储器和互连的面积，以及所需的解码器复制数量。

4) 能耗节省：我们考虑批次为 1 的解码，其中量化权重在每个生成的词元期间从动态随机存取存储器中流式读取一次。在相同困惑度下，更少的每权重比特数会减少存储需求，进而减少权重流量。在单位传输比特能耗模型保持不变的情况下，这将带来成比例的权重传输能耗节省（附录 D-F）。

VI. 结论

我们提出了 BITSHARE，这是一种按组划分的权重格式，将敏感度感知分箱、静态位共享与最优位值分配相结合，在无需码本或微调的情况下，改善大语言模型权重量化中的质量与存储权衡。在 Pile 上进行校准且存储量相同的情况下，BITSHARE 的两种模式在采用相同范围搜索时，其困惑度均低于 GPTQ；在 2 比特下，与 SliM-LLM 相比，BITSHARE-N 可将 WikiText-2 困惑度最多降低 70%，与 GPTAQ 相比最多降低 35%。在达到 SliM-LLM 的 2 比特困惑度时，BITSHARE-U 每权重最多可减少 19.4% 的比特数。解码器的硬件实现验证了其运行的正确性和实际可行性，同时刻画了其适度的开销。未来工作将包括测量图形处理器上的端到端解码速度、功耗分析，以及将位共享与基于梯度的校准相结合。

参考文献

[1] J. Lin、J. Tang、H. Tang、S. Yang、W.-M. Chen、W.-C. Wang、G. Xiao、X. Dang、C. Gan 和 S. Han，《AWQ：面向激活值的权重量化，用于端侧大语言模型压缩与加速》，载于机器学习与系统（*MLSys*）论文集，2024 年。[2] E. Frantar、S. Ashkboos、T. Hoefler 和 D. Alistarh，《GPTQ：生成式预训练变换器的准确训练后量化》，载于国际学习表征会议（*ICLR*），2023年。[3] Y. Li、R. Yin、D. Lee、S. Xiao 和 P. Panda，《GPTAQ：面向非对称校准的高效免微调量化》，载于第*42*届国际机器学习会议（*ICML*）论文集，PMLR 系列，第267卷，2025年，第36 690–36 706页。[4] W. Shao、M. Chen、Z. Zhang、P. Xu、L. Zhao、Z. Li、K. Zhang、P. Gao、Y. Qiao 和 P. Luo，《OmniQuant：面向大语言模型的全向校准量化》，载于国际学习表征会议（*ICLR*），2024 年。[5] Y. Ma、H. Li、X. Zheng、F. Ling、X. Xiao、R. Wang、S. Wen、F. Chao 和 R. Ji，《AffineQuant：面向大语言模型的仿射变换量化》，载于国际学习表征会议（*ICLR*），2024 年。[6] C. Lee、J. Jin、T. Kim、H. Kim 和 E. Park，《OWQ：面向大语言模型高效微调与推理的离群值感知权重量化》，载于*AAAI*人工智能会议论文集，2024 年。[7] T. Dettmers、R. Svirschevski、V. Egiazarian、D. Kuznedelev、E. Frantar、S. Ashkboos、A. Borzunov、T. Hoefler 和 D. Alistarh，《SpQR：一种近乎无损的大语言模型权重压缩稀疏量化表示》，载于国际学习表征会议（*ICLR*），2024年。[8] Z. Yuan、Y. Shang 和 Z. Dong，《PB-LLM：部分二值化大语言模型》，载于国际学习表征会议（*ICLR*），2024 年。[9] W. Huang、H. Qin、Y. Liu、Y. Li、Q. Liu、X. Liu、L. Benini、M. Magno、S. Zhang 和 X. Qi，《SliM-LLM：显著性驱动的混合精度量化》，载于第*42*届国际机器学习会议（*ICML*）论文集，PMLR 系列，第267卷，2025年，第25 672–25 692页。

[10] A. Tseng、J. Chee、Q. Sun、V. Kuleshov 和 C. De Sa，《QuIP#：借助 Hadamard 非相干性和格码本实现更优的大语言模型量化》，载于第41届国际机器学习会议（ICML），2024 年。[11] V. Egiazarian、A. Panferov、D. Kuznedelev、E. Frantar、A. Babenko 和 D. Alistarh，《通过加性量化实现大语言模型的极致压缩》，载于第41届国际机器学习会议（ICML），2024 年。[12] A. Tseng、Q. Sun、D. Hou 和 C. De Sa，《QTIP：借助网格与非相干处理实现量化》，载于神经信息处理系统进展（NeurIPS），2024 年。[13] Y. Chen、J. Meng、J.-s. Seo 和 M. S. Abdelfattah，《BBS：面向深度学习加速的双向位级稀疏性》，载于第57届 IEEE/ACM 国际微体系结构研讨会（MICRO），2024 年，第551–564页。[14] W. Yang、S. Zhou、Y. Nie、Q. Zhou、H. Xu、Z. Li 和 C. Chen，《一种基于列相似性的位级权重重排序策略，用于探索基于 RRAM 的神经网络加速器中的权重稀疏性》，载于 IEEE 第31届并行与分布式系统国际会议（ICPADS），2025年。[15] B. Hassibi、D. G. Stork 和 G. J. Wolff，《最优脑外科手术与通用网络剪枝》，载于 IEEE 神经网络国际会议，1993 年。[16] E. Frantar、S. P. Singh 和 D. Alistarh，《最优脑压缩：准确训练后量化与剪枝框架》，载于神经信息处理系统进展（NeurIPS），第35卷，2022年，第4475–4488页。[17] S. Ashkboos、A. Mohtashami、M. L. Croci、B. Li、P. Cameron、M. Jaggi、D. Alistarh、T. Hoefler 和 J. Hensman，《QuaRot：旋转大语言模型中的无离群值 4 位推理》，载于神经信息处理系统进展（NeurIPS），第37卷，2024 年，第100 213–100 240页。[18] S. Kim、C. Hooper、A. Gholami、Z. Dong、X. Li、S. Shen、M. W. Mahoney 和 K. Keutzer，《SqueezeLLM：稠密与稀疏量化》，载于第41届国际机器学习会议（ICML），2024 年。[19] Z. Dong、Z. Yao、A. Gholami、M. W. Mahoney 和 K. Keutzer，《HAWQ：具有混合精度的神经网络海森矩阵感知量化》，载于 IEEE/CVF 国际计算机视觉会议（ICCV）论文集，2019 年，第293–302页。[20] P. Judd、J. Albericio、T. Hetherington、T. M. Aamodt 和 A. Moshovos，《Stripes：位串行深度神经网络计算》，载于第49届 IEEE/ACM 国际微体系结构研讨会（MICRO），2016 年，第1–12页。[21] J. Albericio、A. Delm´as、P. Judd、S. Sharify、G. O'Leary、R. Genov 和 A. Moshovos，《Bit-pragmatic 深度神经网络计算》，载于第50届 IEEE/ACM 国际微体系结构研讨会（MICRO）论文集，2017 年，第382–394页。[22] H. Lu、L. Chang、C. Li、Z. Zhu、S. Lu、Y. Liu 和 M. Zhang，《为通用深度学习加速提炼位级稀疏并行性》，载于第54届 IEEE/ACM 国际微体系结构研讨会（MICRO），2021 年，第963–976页。[23] S. Lee、D. Ha、S. Kim、S. Kim、H. Lee 和 W. W. Ro，《BitL：用于缩短关键路径的混合位串行与并行深度学习加速器》，载于第58届 IEEE/ACM 国际微体系结构研讨会（MICRO），2025年，第1565–1578页。[24] K. Ibrahim、M. Maheronnaghsh 和 A. Moshovos，《Shannonic：面向机器学习工作负载的高效熵最优压缩》，载于机器学习与系统（MLSys）论文集，2026年。[25] B. D. Rouhani、R. Zhao、A. More、M. Hall、A. Khodamoradi、S. Deng 等，《用于深度学习的微缩放数据格式》，arXiv 预印本 arXiv:2310.10537，2023年。[26] Microsoft，《BC7 格式》，Direct3D 11 文档，
h
t
t
p
s://learn.
microsoft.com/en-us/windows/win32/direct3d11/bc7-format，2024年，访问日期：2026 年 10 月 29 日。[27] S. Banerjee、J. Mathew、D. K. Pradhan 和 S. P. Mohanty，《面向高故障覆盖率的布局感知 Illinois Scan 设计》，载于2010年第11届国际高质量电子设计研讨会（ISQED），2010 年，第683–688页。[28] J. Rajski、J. Tyszer、M. Kassab 和 N. Mukherjee，《嵌入式确定性测试》，IEEE 集成电路与系统计算机辅助设计汇刊，第23卷，第5期，第776–792页，2004 年。[29] M. R. Stan 和 W. P. Burleson，《面向低功耗输入输出的总线反转编码》，IEEE 超大规模集成（VLSI）系统汇刊，第3卷，第1期，第49–58页，1995 年。[30] A. Dasgupta、S. Paria 和 S. Bhunia，《HIPR：通过低开销细粒度删改实现硬件 IP 保护》，IACR 密码学汇刊，*tographic Hardware and Embedded Systems (TCHES)*，第 2025 卷第 3 期，第 781–805 页，2025年。[31] M. Marcellin 和 T. Fischer，“无记忆源和高斯-马尔可夫源的网格编码量化”，*IEEE* 通信汇刊，第 38 卷第 1 期，第 82–93 页，1990 年。[32] H. Touvron、L. Martin、K. Stone、P. Albert、A. Almahairi 等，“Llama 2：开放基础模型与微调聊天模型”，*arXiv* 预印本 *arXiv:2307.09288*，2023年。[33] A. Grattafiori、A. Dubey、A. Jauhri、A. Pandey、A. Kadian 等，“Llama 3 模型系列”，*arXiv* 预印本 *arXiv:2407.21783*，2024 年。[34] A. Yang、B. Yang、B. Zhang、B. Hui、B. Zheng 等，“Qwen2.5 技术报告”，*arXiv* 预印本 *arXiv:2412.15115*，2024 年。[35] E. Frantar、S. Ashkboos、T. Hoefler 和 D. Alistarh，“GPTQ：用于生成式预训练变换器精确训练后量化的代码”，https://github.com/IST-DASLab/gptq，2022，访问日期：2026-08-15。[36] W. Huang、H. Qin、Y. Liu、Y. Li、Q. Liu、X. Liu、L. Benini、M. Magno、S. Zhang 和 X. Qi，“SliM-LLM：面向大语言模型的显著性驱动混合精度量化（代码）”，https://github.com/ Aaronhuang-778/SliM-LLM，2024，访问日期：2026-08-15。[37] Y. Li、R. Yin、D. Lee、S. Xiao 和 P. Panda，“GPTAQ：用于非对称校准的高效免微调量化”，https://github.com/Intelligent-Computing-Lab-Yale/GPTAQ，2025，访问日期：2026-08-15。[38] J. Chee、Y. Cai、V. Kuleshov 和 C. De Sa，“QuIP：具有保证的大语言模型 2位量化”，载于神经信息处理系统进展（*NeurIPS*），2023年。[39] L. Gao、S. Biderman、S. Black、L. Golding、T. Hoppe、C. Foster、J. Phang、H. He、A. Thite、N. Nabeshima、S. Presser 和 C. Leahy，“The Pile：一个用于语言建模的 800GB 多样化文本数据集”，*arXiv* 预印本 *arXiv:2101.00027*，2020。[40] S. Merity、C. Xiong、J. Bradbury 和 R. Socher，“指针哨兵混合模型”，载于国际学习表征会议（*ICLR*），2017年。[41] C. Raffel、N. Shazeer、A. Roberts、K. Lee、S. Narang、M. Matena、Y. Zhou、W. Li 和 P. J. Liu，“探索统一文本到文本变换器的迁移学习极限”，机器学习研究期刊，第 21 卷第 140 期，第 1–67 页，2020。[42] C. Wolf，“Yosys开放综合套件”，https://yosyshq.net/yosys/，2013年。[43] R. Brayton 和 A. Mishchenko，“ABC：一种学术界与工业界适用的验证工具”，载于计算机辅助验证（*CAV*），《计算机科学讲义丛书》第 6174 卷，2010 年，第 24–40 页。[44] Google 和 SkyWater Technology，“SkyWater SKY130 开源工艺设计套件”，https://github.com/google/skywater-pdk，2020年。[45] M. O'Connor、N. Chatterjee、D. Lee、J. Wilson、A. Agrawal、S. W. Keckler 和 W. J. Dally，“细粒度动态随机存取存储器：面向极高带宽系统的节能动态随机存取存储器”，载于*2017* 年第 *50* 届 *IEEE/ACM* 国际微体系结构研讨会（*MICRO*），2017年，第 41–54 页。[46] M. Horowitz，“计算的能耗问题（以及我们能做些什么）”，载于*IEEE* 国际固态电路会议（*ISSCC*），2014 年，第 10–14 页。

附录 A 算法

算法 1 给出了 BITSHARE 对一个权重为 $\mathbf{W} \in \mathbb{R}^{R\times C}$、海森矩阵为 $\mathbf{H}$ 的线性层的编码器，其中海森矩阵由该层的校准输入累积得到。该算法返回布局索引、存储字段、成对比特、组参数以及通道置换 π。

算法 1 用于单个线性层的 BITSHARE 编码器。BITSHARE-U 使用仅含一种布局的菜单运行该算法，因此第 2–5 行 gi遍历每个组 that layout

。要求：权重 W $\in \mathbb{R}^{R\times C}$、由校准输入得到的海森矩阵 H、网格宽度 n、布局菜单 Λ、裁剪因子 P1: $h_j \leftarrow [\mathrm{H}]_{jj}$ 2: 对每个组 g 和布局 $\ell \in \Lambda$ 执行 3: $E_g(\ell) \leftarrow \sum_i \min_{(p,f)\in P\times\{0,1\}} \sum_{j\in g} h_j(w_{ij} - \hat{}w_{ij}^{(\ell)})^2$，使用舍入到格式码 4: 结束循环 5: 根据公式 5 选择 $(\ell_g)_g$ （对$\sum_g \delta(\ell_g)$进行动态规划） 6: 根据 h 和布局构建 π （第 III-B 节）；按照 π 置换 W 的列以及 H 的行和列 7: 对每个组行 (i, g)执行 8: $(s_{ig}, z_{ig}, f_{ig}) \leftarrow$选择 $(p, f) \in P\times\{0,1\}$，使其在布局 ℓ_g的舍入到格式码下具有最小的 $\sum_j h_j(w_{ij} - \hat{}w_{ij})^2$ 9: 结束循环 10: 按 $\max_{j\in\text{unit}} h_j$降序排列量化单元 $\leftarrow$ ：低位箱对（ B的块）以及所有其他单列 11: L $\leftarrow$ 为 $(\mathrm{H} + \lambda\mathrm{I})^{-1}$ 按单元顺序 $\lambda = 0.01\,h$的上三角乔列斯基因子； $d_j \leftarrow L_{jj}$12: 按顺序遍历每个单元 u 13: 如果 u 是列 j，则 14: 对每一行i，取 j区间内允许的最近编码$q_{ij} \leftarrow$ 15: 否则 16: 对 $c \in \{0,1\}$ 执行 17: 依次使用 $b_1 = c$舍入该单元的各列，并将每列的误差传播到下一列 18: 对每一行 i，计算 $J_i(c) \leftarrow \sum_{j\in u}(w_{ij} - \hat{}w_{ij})^2/d_j^2$ 19: 结束循环 20: 对每一行 i，取 $c_{igk} \leftarrow \arg\min_c J_i(c)$ 并保留其编码 21: 结束条件判断 22: 通过 L 将该单元的误差 $(w_{ij} - \hat{}w_{ij})/d_j$ 传播到剩余列 23: 结束循环 24: 返回布局索引、字段、成对比特、 s、 z、 f 和 π

附录 B 格式和编码器细节

该标志位。 该标志位修正了较窄网格的偏差。奇偶性相同的2^{n-1} 编码覆盖组的范围，但一端会缺少一个细步长：奇偶性为 0 的编码缺少顶端编码$2^n - 1$，奇偶性为 1 的编码缺少编码 0。如果将 b_0 与解码器已经持有的某个值（例如零点的奇偶性）绑定，

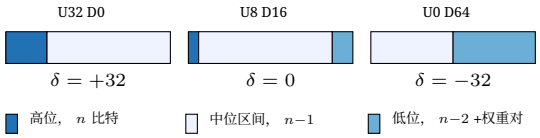


图 5. BITSHARE-N 中一行的三组，每组采用不同布局，各有 $G = 128$ 个位置。在 BITSHARE-U 中，每组具有相同的布局。每组行比特数与参考值相差 $\delta = U - D/2$，且在同一层内，偏移量之和至多为零。该层的每一行都采用相同的布局。

则未覆盖的一端将通过 z_{ig}遵循组的符号平衡，而不是取决于其重要权重所在的位置。该标志位使编码器能够选择这一端，同时也选择低位区间网格的偏移量，每个组行占用 1 位。等价地说，该标志位由组行中的所有 $M_g + D_g$ 中位区间和低位区间权重共享。在 INT4 网格上，中位区间随后成为一个 3 位网格，其步长为范围的 $2/15$，比 INT3 的 $1/7$细 6.7%；在 INT3 网格上，它是一个 2 位网格，步长为$2/7$，比 INT2 的 $1/3$细 14%。

更低比特率。采用 $\delta < 0$ 的布局存储的比特更少。为了进一步降低比特率，成对比特可推广为由 B 个连续低位区间位置共享的比特，从而得到每组行 D/B 比特和 $\delta = U - D$（ $1 - 1/B$）。在没有这种比特时， b_1 会被丢弃：每个低位区间编码都会从零点的第 1 个比特z_{ig}中取用该比特，且 $\delta = U - D$。两种模式都使用权重对（$B = 2$）。为了将 BITSHARE-U 的率失真曲线向下延伸直至崩溃，IV-D 和 IV-E 节还在每个组中使用 $B \in \{4, 8, 16\}$ 和被丢弃的比特。

布局。 图 5 展示了 BITSHARE-N 菜单的三种布局。这些布局遵循各组的敏感度，但并非完全由敏感度决定：在我们评估的模型中，组均值 h_j 与其偏移量之间的秩相关为 $+0.3$ 到 $+0.7$ 。

舍入。 对于由 B 个低位区间位置共享的比特，将由 B 列组成的模块像权重对一样进行舍入，一列接一列地处理。权重始终舍入到与共享比特一致的最近编码，这也可能改变其存储字段（产生进位或借位），而不是覆盖该权重自身的共享比特。

编码器的额外工作。编码器无需微调，也无需码本。与采用相同范围搜索的 GPTQ 相比，它的额外工作是在列循环开始前对菜单中的每种布局进行范围搜索（在 BITSHARE-N 中），在该搜索中使用第二个标志位取值，并对每个低位区间权重对和每一行执行两种候选舍入。

附录 C 附加结果

表 VII 在 WikiText-2 上重复了表 IV 的设置，窗口大小为 4096 个词元。表 VIII 在 SliM-LLM 的协议下，将 BITSHARE 的两种模式与我们运行的 SliM-LLM 和 GPTAQ 编码进行了比较，涵盖全部三个测试集。

在表 VIII 的全部三个测试集上，BITSHARE-N 在 2 比特时比 SliM-LLM 低 29%–93%，在 3 比特时低 0.9%–10.5%。在 2 比特时，它也比 GPTAQ 低 14%–98%。在 3 比特时，它在 C4 上比 GPTAQ 低 2.5%–5.5%，在 WikiText-2 上低 0.9%–3.8%。

对于 Llama-3-8B，而在 WikiText-2 上，对于 Llama-2，GPTAQ 的在四个单元中的三个单元里更低，最多低 0.4%。

表 VII WikiText-2 上使用 4096-token 窗口时的困惑度，校准在 Pile 上进行（表 IV 中的设置）。每种位宽对应的最低困惑度以 **粗体** 标出。

		Llama-2		Llama-3		Qwen2.5	
方法	b/w	7B	13B	8B	0.5B	1.5B	3B
FP16	16	5.12	4.57	5.75	12.08	8.57	7.45
3 比特							
GPTQ	3.148	5.82	5.05	7.42	18.06	11.15	9.22
GPTQ + 裁剪	3.148	5.67	5.10	7.32	17.60	10.40	8.76
BITSHARE-U	3.164	**5.62**	5.00	6.95	16.18	**10.10**	8.60
BITSHARE-N	3.164	5.63	**4.95**	**6.93**	**15.92**	10.11	**8.56**
2 比特							
GPTQ + 裁剪	2.141	9.83	7.75	39.38	109.7	30.12	19.75
G=64	2.281	8.72	7.15	33.80	**50.91**	22.85	15.77
BITSHARE-U	2.156	8.62	6.98	16.01	57.84	**18.90**	14.19
BITSHARE-N	2.156	**8.38**	**6.79**	**13.75**	53.83	18.96	**14.09**

A. 基线运行

SliM-LLM。在 Llama-2-70B 的一个线性层上，其显著性计算在 Cholesky 分解中失败，因此我们使用更大的阻尼重新进行计算。按照已发布的位宽，我们的运行结果在 3 比特时与 SliM-LLM 报告的困惑度相差不超过 1.4%，在 2 比特时则高出 0.6%–11.8%。在 70B 模型上，这些结果从低于其 6.9% 到高于其 37% 不等。

GPTAQ. 在 3 比特下，我们分别采用量化前固定组以及按 act-order 形成的组运行该方法，并报告每个模型上表现较好的结果。在 2 比特下，由于第二种方式在 Llama-2-7B 上出现分歧，因此采用固定组。

B. 重复运行

在 2 比特下，运行结果取决于浮点舍入。在 RTX 6000 Ada GPU 上而非 RTX A6000 上运行时，BITSHARE-U 在 Llama-2-7B 上的结果为 8.78，而非 10.53。两种图形处理器对第一个解码器层输出的舍入结果不同。因此，其查询投影的海森矩阵相对差异为$8\cdot10^{-7}$，而 GPTQ 已有 0.16% 的权重得到不同的 2 位编码。在同一种图形处理器上，运行结果具有确定性。表 IX 重复了 GPTQ + 裁剪、BITSHARE-U 和 BITSHARE-N 在 Llama-2-7B 和 Llama-3-8B 上的 2 比特运行：两次将校准序列换序（这只会改变对这些序列求和时的舍入结果），另两次则使用其他序列。在五次运行中，BITSHARE-U 和 BITSHARE-N 的困惑度对数标准差为 0.05–0.06；GPTQ + 裁剪在 Llama-2-7B 和 Llama-3-8B 上分别为 0.03 和 0.22。所有十次运行中，BITSHARE-N 的结果都低于 GPTQ + 裁剪，平均分别低 8.4% 和 42%；两种模式的每次运行结果也都远低于 SliM-LLM 的编码（17.31 和 44.35）。与 BITSHARE-U 相比，BITSHARE-N 在 Llama-2-7B 的五次运行中有三次结果更低，在 Llama-3-8B 上则有两次。平均而言，它在 Llama-2-7B 上低 1.3%，在

Llama-3-8B。在这两个模型上，按照 SliM-LLM 的协议，在 2 比特时 BITSHARE-N 相较于 BITSHARE-U 的增益处于不同运行之间的波动范围内。

C. 直至崩溃

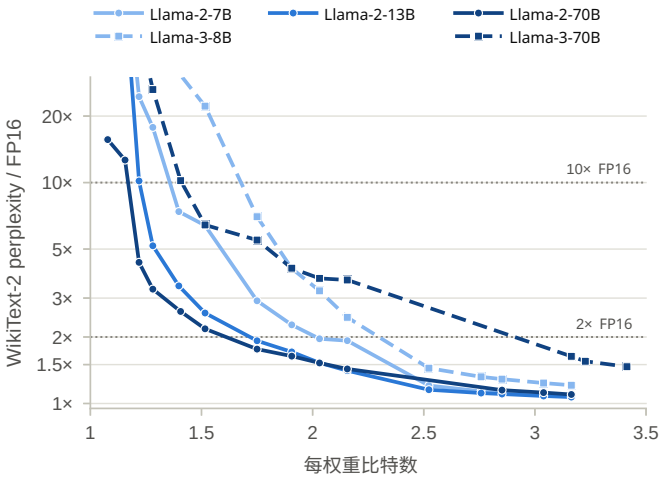


图6. 在 SliM-LLM 的协议下，BITSHARE-U 的率失真曲线：WikiText-2 困惑度（2048令牌窗口）相对于 FP16 的变化与每权重比特数的关系，并标出每种速率下的最佳格式。浅色线表示较小的模型；虚线表示 Llama-3。点线表示 FP16 的 2 倍和 10 倍（表 X）。

曲线定律。对于 $d = \ln(\text{ppl}/\text{ppl}_{\text{FP16}})$，在不超过 FP16 的三倍范围内，每条曲线都符合 $\ln d = a_m + \beta_m b$，其中 b 表示每权重比特数。（表 X）显示，同一家族内的斜率随模型规模增大而趋于平缓；一个共同斜率 $\beta = -1.59$，加上每个模型各自的一个截距，可拟合全部 37 个数据点，$\ln d$上的误差为 0.17。因此，只需运行一次模型，就能预测其曲线：固定 a_m（取最接近低于 FP16 三倍范围内每权重 2.156 比特的那次运行），并从其他模型中获取 β，即可预测其余 32 个数据点；困惑度的中位误差为 5.2%，与 2 比特时不同运行之间的波动幅度相当（标准差为 5–7%，见附录 C-B）。最大误差为 48%，出现在 Llama-2-70B 上；它是斜率最平缓的 Llama-2 模型，在每权重 1.406 比特时出现该误差。这一定律仅在不超过 FP16 三倍的范围内成立。在我们以低于每权重 1.516 比特的设置运行 70 B 模型之前，外推结果显示，在 Llama-2-70B 上达到十倍 FP16 需每权重 0.6–0.9 比特，而该模型实际在 1.17 时达到；在 Llama-3-70B 上则需 1.2–1.6 比特，而该模型实际在 1.41 时达到。

D. 消融实验

分箱。不使用分箱（U0 D0）时，组内每个权重都存储在中位区间的网格上，并带有标志位。在 Llama-2 上，其结果比 BITSHARE-U 高出不到 0.6%；但在 Llama-3-8B 和通义千问模型上，则高出 8–65%。在这些模型中，每组有少数几个通道需要使用高位区间的完整 INT3 网格。BITSHARE-N 的菜单包含 U0 D0，且在所有模型上的表现都优于二者。

标志位。根据零点的奇偶性，从中位区间和低位区间提取 b_0 位可节省标志位，每个权重节省 1/128 位，但六个模型的困惑度会提高 5–30%；在 Llama-2-7B、Llama-2-13B 和 Llama-3-8B 上，困惑度分别提高 13%、16 % 和 30%。

组间分配。每个组的比特数相等时，每个组只能改变其布局的形状（$U = D/2$）。与 BITSHARE-U 相比，这使困惑度平均降低了 2.6%，但在 Llama-3-8B 和 Qwen2.5-1.5B 上分别升高了 6.1% 和 1.8%。在三个模型上，它低于 BITSHARE-N，幅度为

表 VIII 在 SliM-LLM 的协议（在 WikiText-2 上进行校准）下的困惑度，与我们运行采用其发布的组宽度的 SliM-LLM 编码及 GPTAQ 编码所得结果的对比。W2：使用 2048 或 4096-token 窗口的 WikiText-2。每种宽度下最低的困惑度以 **粗体**显示。

		Llama-2-7B			Llama-2-13B			Llama-3-8B		
方法	b/w	W2 2048	W2 4096	C4	W2 2048	W2 4096	C4	W2 2048	W2 4096	C4
FP16	16	5.47	5.12	7.26	4.88	4.57	6.73	6.14	5.75	9.44
3 比特										
SliM-LLM（编码）	3.148	6.33	5.85	8.55	5.26	4.92	7.50	7.69	7.16	13.74
GPTAQ（编码）	3.148	5.93	**5.53**	8.70	**5.19**	**4.86**	7.55	7.37	7.11	12.74
GPTQ + 裁剪	3.148	5.96	5.56	8.30	5.33	4.99	7.58	7.34	6.88	12.74
BITSHARE-U	3.164	5.94	5.54	**8.22**	5.22	4.89	7.38	7.42	6.94	12.38
BITSHARE-N	3.164	**5.92**	5.54	**8.22**	5.21	4.87	**7.36**	**7.31**	**6.85**	**12.29**
2 比特										
SliM-LLM（编码）	2.141	17.31	19.67	64.27	9.47	8.78	18.79	41.83	44.35	434.7
GPTAQ（编码）	2.141	11.76	13.64	28.93	7.84	7.46	16.57	20.87	18.76	1.2e3
GPTQ + 裁剪	2.141	10.37	9.65	**17.79**	7.41	6.84	12.02	21.95	21.22	228.6
BITSHARE-U	2.156	10.53	9.75	18.27	6.88	6.40	10.81	15.06	13.87	33.65
BITSHARE-N	2.156	**10.11**	**9.25**	18.72	**6.70**	**6.25**	**10.46**	**14.47**	**13.29**	**30.81**

表 IX

SliM-LLM 协议下 AT2 位的重复运行：表 III 中所示运行（RUN）的 WikiText-2 困惑度；使用相同运行、但将 128 个校准序列按另外两种顺序排列（ORDER），以及采用另外两组序列抽样（SEED）时的困惑度。新的顺序只会改变对序列求和时的舍入。窗口长度与表 III 相同：Llama-2-7B 为 2048 个词元，Llama-3-8B 为 4096 个词元。MEAN：几何平均值；SD：五次运行中对数的标准差。

方法	运行顺序 1	顺序 2	随机种子 1	随机种子 2	平均值	sd
Llama-2-7B						
GPTQ +裁剪	10.37	10.40	10.43	9.86	9.87	10.18 0.029
BITSHARE-U	10.53	9.40	9.18	8.91	9.34	9.46 0.064
BITSHARE-N	10.11	9.14	9.24	9.04	9.16	9.33 0.046
Llama-3-8B						
GPTQ + 裁剪	21.22	20.89	26.37	22.72	35.64	24.84 0.222
BITSHARE-U	13.87	14.98	14.85	14.02	13.12	14.15 0.054
BITSHARE-N	13.29	15.03	13.85	14.76	14.59	14.29 0.051

表 X

直至前述 BITSHARE 在 SliM-LLM 的协议下，U（每个组采用一种布局）：WikiText-2 困惑度达到 FP16 的 2 倍和 10 倍时的速率（每权重比特数），其中每种速率均采用评估得出的最佳格式。 β_m：在达到 FP16 的 3 倍之前，$\ln d$ 相对于每权重比特数的斜率，其中 $d = \ln(\mathrm{ppl}/\mathrm{ppl}_{\mathrm{FP16}})$。1.281 B/W：在 1.281 每权重比特数下的困惑度，此时比特 b_1 由八个低分箱权重共享（U0 D128），且 b_1 被丢弃（U8 D120）。

	× FP16			1.281 位/权重	
模型	2	10	β_m	共享	丢弃
Llama-2-7B	2.02	1.36	−1.97	97.30	199.8
Llama-2-13B	1.72	1.22	−1.69	25.28	67.50
Llama-2-70B	1.61	1.17	−1.30	10.93	18.94
Llama-3-8B	2.30	1.68	−1.50	904.7	2.9e4
Llama-3-70B	2.91	1.41	−0.93	75.27	87.33

在通义千问 Qwen2.5-0.5B 上最高可达 7.9%。像 BITSHARE-N 那样依据公式 5 在组之间移动比特，可使所有模型的这一数值降低，平均降低 3.6%。获得比特的组更为敏感：在六个模型中，组的均值 h_j 与其偏移量之间的秩相关介于 $+0.38$ 和 $+0.66$ 之间。不同模型系列的布局也有所不同。在 Llama-

表 XI

2 比特下的消融实验 1 BITSHARE-U 2 + 裁剪：GPTQ 在 INT2 网格上的 + 裁剪，作为起点（表 IV）。无分箱：每组均为 U0 D0。TIED b_0：U8 D16， b_0 比特取自零点的奇偶性，而非标志位。BITSHARE-U：U8 D16。等比特：每组采用一种布局，每组包含 $\delta = 0$ 。BITSHARE-N：按公式 5 为每组设置布局。每个模型的最低困惑度以 **粗体**显示。

		Llama-2		Llama-3	Qwen2.5		
方法	b/w	7B	13B	8B	0.5B	1.5B	3B
INT2 +裁剪	2.141	10.49	8.31	39.51	129.9	32.87	21.74
无分箱	2.156	9.30	7.39	18.32	105.2	24.26	16.70
Tied b_0	2.148	10.42	8.61	21.18	66.73	24.98	17.92
BITSHARE-U	2.156	9.25	7.43	16.35	63.63	20.61	15.44
等比特	2.156	9.04	**7.19**	17.34	**53.46**	20.98	**15.17**
BITSHARE-N	2.156	**8.95**	7.31	**15.19**	58.02	**20.60**	15.29

2，在分箱贡献较小的情况下，最常见的布局最多包含四个高位区间位置（在 Llama-2-7B 上，U4 D8 占 17% 的组，U4 D0 占 17%，U0 D0 占 14%）；而在 Qwen模型中，最常见的布局有 16 或 24 个（在 Qwen2.5-0.5B 上，U24 D48 占 14%，U24 D32 占 10%，U16 D32 占 9%）。图 7 展示了 BITSHARE- N 为 Llama-2-7B 第 2 层和第 10 层解码器层的输出投影选择的布局，其中每组对应一个注意力头。敏感度较高的组采用更大的高位区间，敏感度较低的组则采用更大的低位区间，组的均值 h_j 与其偏移量之间的秩相关为 $+0.70$ 和 $+0.92$ 。这两层的 32 个组分别使用了 19 种和 14 种不同的布局，而同一层的查询投影则分别使用 8 种和 4 种。

附录D

硬件实现细节

A. 微体系结构

存储。组行按照行主序首尾相接存储，不进行填充，因此组行可以从任意比特开始。每条记录按照解码器读取的顺序包含表II中的字段：缩放因子 s_{ig}、零点 z_{ig} 和标志位 f_{ig}，然后是按位置顺序排列的每个权重的字段，最后是 $D_g/2$权重对

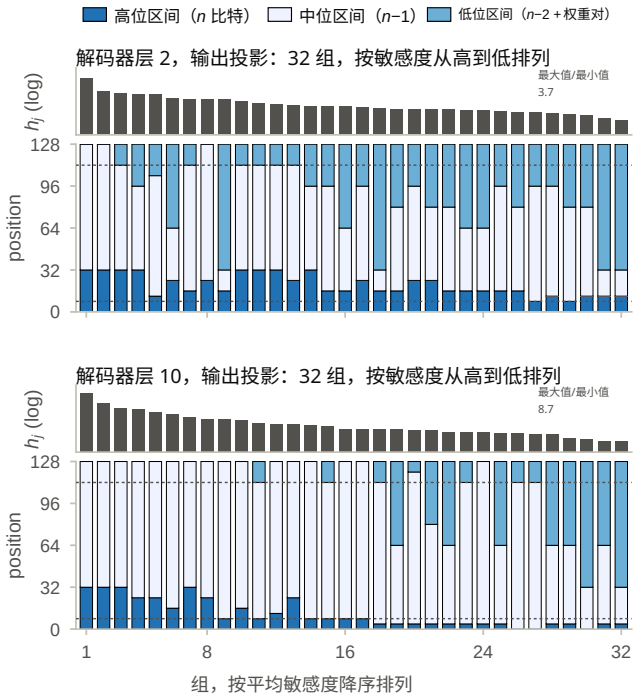


图 7. BITSHARE-N 在 2 比特下为 Llama-2-7B 的解码器层 2 和 10 的输出投影所选择的分箱配置（Pile 校准）。其 32 个组各包含 128 个输入通道，每组对应一个注意力头，并按平均敏感度 h_j（灰色条带，对数刻度）排序。条形图显示每个组的高位、中位区间和低位区间位置，如图 5 所示。虚线：U8 D16，即 BITSHARE-U 的布局。

比特。一条记录包含 $B_g = (n-1)G + \delta(U_g,\ D_g) + n + 17$ 比特（公式 3）。

层解码器。逐层表为每个组保存一个 6 位布局索引，该索引由所有 R 行共享。一个包含 44 个条目的只读存储器将该索引映射为（U_g，D_g），进而映射为 B_g；对于第 IV-D 节中的每层单一布局变体，只读存储器由固定的（U，D）旁路，其中 b_1 由两个或四个低位区间权重共享。遍历器按照存储顺序请求组行，位对齐器从由 W位存储器字填充的寄存器池中截取每个 B_g 位记录。由于每行都具有相同的布局序列，遍历器无需存储偏移量；对于随机访问，前缀和单元返回每行内每个组的比特偏移量。保留索引和非法布局会引发误差并停止解码。

组解码器。组解码器（图 4(a)中的点状框）在每个时钟周期输出 L 个权重，位于一个组行的连续位置t ＝基址 . . . ，基址$+L-1$；我们将这样一次输出传输称为一次拍。根据第 III-E 节，通道通过两次比较确定其位置所在的区间，$t < U_g$ 和$t \geq G - D_g$。其字段从拍内的第 $n\,h+$（$n-1$）$m+$（$n-2$）l 个比特开始，其中 h、m 和 l 分别统计［基址 t）中的高位、中位和低位位置。随后，该通道根据公式 2 形成 q，其中 f_{ig} 对于 $t \geq U_g$ 为 b_0，而成对比特 c_{igk} 在低位区间中为 b_1。三个寄存级计算 $q - z_{ig}$，即 $|q - z_{ig}|$ 与 s_{ig} 的有效数的精确整数乘积，并执行一次舍入到最近偶数至 binary16 的操作；这就是按照 INT-n 完全相同的方式精确计算公式 1。对于在整数中进行累加并应用 s_{ig}，每个组行一次。拍可能跨越区间边界，因此无论布局如何，每个组行都恰好需要 G/L 拍。

吞吐量。对于 $G/L \geq 2$，解码器可维持 $\min(L,\ WG/B)$ 每周期权重数，其中 B 为平均记录大小：当内存接口能够跟上时为 L，否则为输入带宽。借助 $G = 128$ 和 $L = 16$，对于菜单中的每种布局，64 比特接口都足够；在 $n = 4$ 时每周期最多需要 54.6 比特，在 $n = 3$时需要 38.5 比特。在仿真中，对于两个网格，包含全部 44 种布局的行在 $L \in \{4, 8, 16\}$ 时均可恰好以每周期一拍的速度完成解码。在 $L = 32$且采用 $W = 64$时，对于 $n = 3$ 和 $n = 4$处的 352 拍，分别需要 369 和 544 个周期。

置换。解码器按区间顺序输出权重，因此激活值通过第 III-B 节中的置换 π 读取。由于 π 由一层的所有 R 行共享，一个激活值向量每个词元和每层只需置换一次，而不必每行置换一次。因此，我们报告的解码器不包含激活值收集。

B. 存储布局

- 一层按行存储；每行由其 C/G 组行串接而成；不进行填充。数据流按最低有效位优先：第一个存储字的第 0 位就是该层的第 0 位。
- 组行按以下顺序排列：s_{ig}（16 比特，IEEE binary16）；z_{ig}（n 比特）；f_{ig}（1 比特）；U_g、M_g 和 D_g字段，分别包含 n、$n-1$ 和 $n-2$ 比特，并按位置顺序排列；以及 D_g/S 共享比特，采用 $S = 2$，或在第 IV-D 节的四路变体中采用 $S = 4$。每个字段均按最低有效位优先排列。
- 布局索引 $k \in \{0, \ldots, 43\}$ 按字典序（U，D）枚举菜单 Λ；索引 44-63 保留。

SliM-LLM解码器存储 s（16 比特）、z（$N+1$ 比特，足以容纳最大精度）以及该组精度对应的 G字段；INT-n 解码器存储 s、z（n 比特）、一个未使用的比特，以及 n比特的 G字段。

C. 验证

我们使用 Python参考模型对层解码器进行验证。该模型采用串行位游标解析打包位流，独立于寄存器传输级中的逐通道偏移量计算。模型通过对精确乘积 $s_{ig}(q - z_{ig})$ 进行一次舍入，重建 binary16 权重。在 62 次仿真中，回归测试检查每个输出位和旁带信号，涵盖：

- 两种量化网格和通道数 $L \in \{4, 8, 16, 32\}$；
- 全部 44 种布局，以及覆盖 86 个组和三行的随机逐组布局，与 Llama-2-7B 下投影的输入组数量相匹配；
- 带有 $D \in \{16, 48, 80, 96, 120\}$ 的 U8 布局，包括二路和四路共享、退化布局以及组大小 $G = 64$；
- 个 32、64 和 128 比特的存储字、随机输入间隔以及两个连续层。

表 XII 每个模块的逻辑综合（YOSYS 0.62 和 ABC、SKYWATER SKY130_FD_SC_HD 子集、TT 025C 1V80、$G = 128$、$L = 16$、$W = 64$）。逻辑延迟不包括寄存器时序和互连。

模块	面积（μm^2）	单元	触发器	逻辑延迟（ns）
BITSHARE 组解码器，$n = 4$	210,243	25,264	2,327	7.88
BITSHARE 组解码器，$n = 3$	175,384	20,113	1,957	7.63
SliM-LLM 组解码器，$\{2, 3, 4\}$	129,728	13,464	2,169	7.36
SliM-LLM 组解码器，$\{1, 2, 3\}$	105,057	11,154	1,799	6.13
INT3 组解码器	100,693	10,095	1,795	6.39
INT2 组解码器	80,106	8,184	1,425	6.15
位对齐器，记录 $\leq$ 533 比特（BITSHARE，$n = 4$）	107,557	15,883	1,184	5.03
位对齐器，$\leq$ 532 比特（SliM-LLM，$\{2, 3, 4\}$）	107,489	15,817	1,182	4.96
位对齐器，$\leq$ 404 比特（BITSHARE，$n = 3$；INT3）	82,593	11,413	924	5.47
位对齐器，$\leq$ 403 比特（SliM-LLM，$\{1, 2, 3\}$）	83,615	11,866	922	4.90
位对齐器，$\leq$ 275比特（INT2）	59,822	8,397	666	4.84
布局只读存储器（44 个条目）	334	66	0	1.14
Binary16 反量化器，单通道	2,110	352	0	5.91
BITSHARE 层解码器，$n = 4^a$	381,838	42,257	5,320	8.29
BITSHARE 层解码器，$n = 3^a$	329,181	36,019	4,689	7.45

[a]组解码器、对齐器、只读存储器、遍历器，以及一个存储在触发器中的 256 项布局表；部署时，该表应存放在静态随机存取存储器中。

D. 各模块的综合结果

表 XII 列出了硬件实现中合成的每个模块：BITSHARE、SliM-LLM 和 INT-n在两种速率下的组解码器、每种格式的位对齐器、布局只读存储器、一个 binary16 反量化通道以及完整的 BITSHARE 层解码器。在 3 比特和 2 比特下，BITSHARE 组解码器的面积分别是 SliM-LLM 解码器的 1.62× 倍和 1.67× 倍，触发器数量多出 7–9%，因此额外面积大部分来自逻辑电路。BITSHARE 和 SliM-LLM 的位对齐器面积几乎相同（3 比特下为 107.5kμm^2），而布局只读存储器仅增加 334 μm^2。

E. 相同困惑度下的权重流量

对于受内存带宽限制的批次为 1 的解码，每秒令牌数受每个词元所需流式传输的字节数限制。表 XIII 将表 V 中的存储空间缩减量转换为这一上限：在相同困惑度下，SliM-LLM（运行我们所用的已发布代码）与 BITSHARE 每个词元的字节数之比；计算时纳入量化线性层和 FP16 输出层，但不包括键值缓存。该上限沿用表 V 的假设：采用第 IV-D 节所述的每层单一布局配置，并按其速率—质量曲线插值得出速率。这些是上限，而非实测加速比；由于解码器能够跟上内存速度，且各行的每组解码工作量均匀，融合内核有望接近这些上限，未来工作将对此进行探索。

F. 相同困惑度下的权重流量能耗

在批次为 1 的解码中，每个量化权重每个词元都会从 DRAM 读取一次，因此权重流量的能耗为$E = N \times b \times e_{\text{bit}}$，其中 N 是量化权重的数量，b 是每权重比特数，e_{bit} 是每传输一比特所需的 DRAM 能耗。N 和 e_{bit} 对所有格式都相同，因此相对节省量为 $1 - b/b_{\text{ref}}$，且不依赖任何假设常数。在存储量相近时，BITSHARE 的每权重比特数为 2.156 比特，而 SliM-LLM 为 2.141，比其高约 0.7%；在这一工作点上，其优势是困惑度更低。表 XIV 将 2 比特下节省的存储量换算为权重能耗，其中 HBM2 [45] 的能耗为 $e_{\text{bit}} \approx 4$ pJ/bit，封装外 DRAM [46]的能耗为 ≈ 20pJ/bit。在 WikiText-2 困惑度相同的情况下，BITSHARE-U 读取的权重比特数比 SliM-LLM 少 13.6–19.4%（在 C4 上至少少 21.6%），使用 HBM2 可为每个权重节省 1.2–1.7 pJ，使用封装外 DRAM 则可节省 5.8–8.3 pJ。在 Llama-2-7B（6.5×10^9 个量化权重）上，这意味着每个词元的权重流量减少 336 MB，即每个词元节省 11–54 mJ。该估算基于分析：仅涵盖权重流量，不包括 KV 缓存、激活值或 FP16 输出头，因为这两种格式在这些方面相同；同时，估算采用了文献中的 e_{bit}。

表 XIII 相同困惑度下相对于 SLIM-LLM 的解码加速比上限。

	3 比特		2 比特	
模型	WikiText-2	C4	WikiText-2	C4
Llama-2-7B	1.15×	1.05×	1.19×	1.30×
Llama-2-13B	1.02×	1.03×	1.20×	1.25×
Llama-3-8B	1.03×	1.08×	1.12×	$\geq$1.23×

表 XIV 相同 2 位 WikiText-2 困惑度下的权重流量能耗：BITSHARE-U 达到 SliM-LLM 的困惑度（每权重 2.141 比特）时的每权重比特数、节省的动态随机存取存储器能耗，以及两种动态随机存取存储器能耗下的每权重节省量。

模型	每权重比特数	节省的能量	每个权重节省的皮焦耳（4 / 20 皮焦耳/比特）
Llama-2-7B	1.726	19.4%	1.66 / 8.31
Llama-2-13B	1.745	18.5%	1.58 / 7.92
Llama-2-70B	1.751	18.2%	1.56 / 7.79
Llama-3-8B	1.758	17.9%	1.53 / 7.66
Llama-3-70B	1.850	13.6%	1.16 / 5.82