文章编号:1671-4598(2026)02-0174-08 DOI:10.16526/j.cnki.11-4762/tp.2026.02.023 中图分类号:TP391 文献标识码:A

基于申威众核架构的层归一化加速与优化

王 鑫,姚柄彤
(江南大学 教育部轻工过程先进控制重点实验室,江苏 无锡 214122)

摘要:针对层归一化在高性能计算中面临的访存密集性问题,提出以申威众核处理器为平台的并行层归一化计算方案;该方案的核心思想是高效利用SW26010P众核处理器的计算资源和传输带宽,通过采用两种不同的分核策略对数据进行划分,结合双缓冲机制、DMA技术和SIMD向量化等优化手段,来实现计算任务的并行化处理;实验测试结果表明,与主核串行算法相比,使用并行层归一化可以获得55.48的最高加速比;与使用SIMD指令优化前的并行层归一化相比,经过SIMD指令对并行层归一化进行数据并行优化的最大有效算力为28.25 GFLOPS。

关键词:高性能计算;申威众核处理器;层归一化;访存密集;并行计算

Acceleration and Optimization of Layer Normalization Based on Sunway Many-core Architecture

WANG Xin, YAO Bingtong
(Key Laboratory of Advanced Process Control for Light Industry (Ministry of Education), Jiangnan University, Wuxi 214122, China)

Abstract: To address the memory access-intensive issue of layer normalization in high-performance computing, this paper proposes a parallel layer normalization computing scheme on a Sunway multi-core processor platform, and its core idea is to efficiently utilize the computational resource and bandwidth of an SW26010P multi-core processor. Two different core distribution strategies are adopted to classify the data, and optimization techniques such as double buffering, DMA technology, and SIMD vectorization are combined to effectively realize the parallel processing of computational tasks. Experimental results demonstrate that, compared to the main core serial algorithm this parallel approach achieves a maximum speedup ratio of 55.48. Compared to the parallel layer normalization without SIMD instruction optimization, the maximum effective computing power acquired by the data parallel optimization of layer normalization with SIMD instructions reaches 28.25 GFLOPS.

Keywords: high-performance computing; Sunway many-core architecture; layer normalization; memory access-intensive; parallel computing

0 引言

随着深度学习[1]技术的飞速发展,高性能计算(HPC, high-performance computing)在大规模神经网络训练与推理过程中扮演着至关重要的角色,在图像分割[2]、自然语言处理[3-4]、计算机视觉[5]等前沿领域均取得了显著成就。但是随着网络结构深度的日益深化和过拟合的风险加剧,梯度消失与梯度爆炸等复杂问题极大地阻碍了模型的训练进程,因而,高效地训练深度神经网络成为当前的一大技术挑战。

归一化(Normalization)技术的提出成功地缓解了梯度爆炸与梯度消失的问题。其中,批量归一化(BN, batch normalization)[6]是一种加速训练的经典技术,但是其数据的均值和方差高度依赖于批量大小的合理设置,所以在批量大小非常小的分布式模型训练场景中,批量归一化的应用会受到限制。并且在处理递归神经网络(RNN, recurrent neural network)[7-8]时,序列长度的动态变化使得批量归一化的效果往往不够理想。为了克服这一问题,深度学习领域中的横纵数据预处理技术层归一化(LN, layer normalization)[9]被提出。层归一

收稿日期:2025-02-05; **修回日期**:2025-03-12。
基金项目:高等学校学科创新引智计划项目(B23008)。
作者简介:王 鑫(1981-),男,博士,讲师。
引用格式:王 鑫,姚柄彤.基于申威众核架构的层归一化加速与优化[J].计算机测量与控制,2026,34(2):174-181.

化通过在每个训练步骤中对神经元的激活值进行归一化处理，使得神经元的激活值在训练过程中保持在一个相对稳定的范围内，从而缓解梯度消失或梯度爆炸问题，提升了模型的训练稳定性和效率。层归一化对批量大小的完全独立性特点使其在批量大小为 1 时也能有效地工作。正是因为该方法的简单性和有效性，层归一化已成为深度模型优化工具中的一大利器，对于稳定训练过程、加速收敛速度以及提升模型性能具有关键作用，在不同任务上的成功应用不仅验证了其普适性和高效性，也推动了深度学习技术的进一步发展[10-12]。但是随着模型规模的日益增大，层归一化的访存密集性成为制约高性能计算平台性能提升的关键因素之一。

超级计算机作为处理复杂计算任务的强大工具，为解决访存密集型和计算密集型的高性能计算问题提供了良好的平台[13-15]。我国的“神威·太湖之光”超级计算机完全采用自主研发技术，以其超过 100×101^{15} 次每秒的峰值浮点运算能力（FLOPS）刷新了世界纪录，成功登顶世界之巅[16]。在 2021 年，由多个 SW26010P 处理器组成的新一代神威超级计算机“神威·太湖之光”发布，SW26010P 众核处理器采用独特的众核架构，集成了大量的计算核心和高效的片上网络，为并行计算提供了坚实的基础。

本文的主要工作是根据 SW26010P 众核处理器的众核架构特点和编程规范，提出众核优化方案，来解决层归一化访存密集性问题。针对测试用例的类型与特性，本文设计了两种不同的分核策略，即泛化性并行设计（GPD，generalized parallel design）和特定场景优化的并行设计（PDSS，parallel design optimized for specific scenarios），旨在通过并行计算提升层归一化的计算效率。从性能上评估，与主核串行算法相比，GPD 优化后的层归一化获得了 55.46 的最高加速比，PDSS 优化后的层归一化获得了 51.19 的最高加速比。通过单指令多数据流（SIMD，single instruction multiple data）的优化，层归一化的最大有效算力达到了 28.25 GFLOPS。

1　硬件架构和基础算法

1.1　SW26010P 众核处理器

SW26010P 众核处理器集成了 6 个核组（CG，core group）[17]，每个核组包含一个运算控制核心（MPE，management processing element），简称主核；和一个 8×8 阵列的运算核心（CPE，computing processing element），简称从核阵列，总计 390 个核心。同一核组的主核和从核阵列共享一个内存控制器（MC，memory controller），每个核组的主核拥有 16 GB 的 DDR4-3200 内存和 51.2 GB/s 的内存带宽，总内存带宽为 307.2 GB/s。通过高效的片上互联网络，该处理器实现从核与主核的交互。其组成结构如图 1 所示。

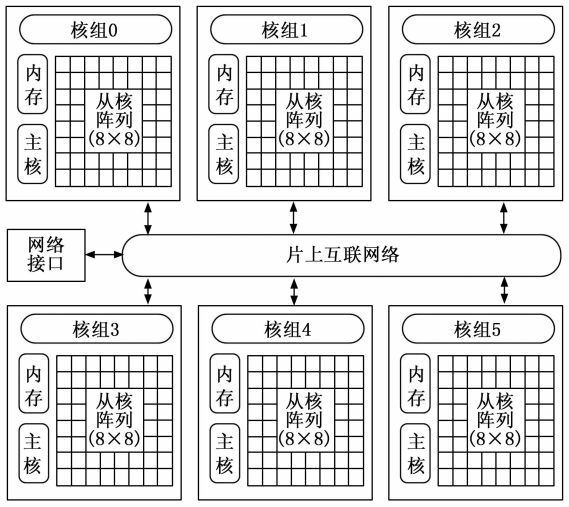

图 1　申威众核处理器 SW26010P 结构图

主核和从核都是完整的 64 位精简指令集计算机（RISC，reduced instruction set computer）内核。其中，主核采用的是自主设计的 SW64 指令集，具有 L1 和 L2 两级缓存结构。64 kB 的 L1 缓存分成了 32 kB 的指令缓存和 32 kB 的数据缓存，L2 高速缓存的大小为 512 kB。主核与从核的频率分别为 2.1 GHz 和 2.25 GHz。主核在系统中主要有控制、通信和 I/O 管理等任务，从核则专注于计算任务。

为了确保数据处理的独立性和高效性，每个从核均配备了 256 kB 的局部数据空间（LDM，local data memory）。LDM 是从核的本地存储结构，主要分为私有空间和连续共享空间。私有空间可以支持当前从核的快速数据访问，连续共享空间主要用于从核间的数据共享，从核可以访问核组内其他从核的共享空间。

LDM 和主存之间的数据传输主要有两种方式。从核可以通过 Load/Store 方式直接使用加载和存储指令访问主存中的数据。这种方式简单直接，适合访问离散数据或小块数据，尤其在对少量数据进行读写操作时表现出较高的灵活性。然而，由于主存的访问速度远低于从核的 LDM，每次访问主存都会产生较大的延迟。LDM 和主存之间的数据传输也可以通过直接存储器访问（DMA，direct memory access）技术[18]，从核阵列内的从核之间可以通过远程读入/写出以及远程内存访问（RMA，remote memory access）技术[19]进行高效的数据通信。RMA 技术通过强化从核间的数据交换能力，实现了低延迟的通信机制，进一步提升了处理器的整体效能。性能表现上，SW26010P 众核处理器的单精度峰值性能为 14.026 TFLOPS，半精度峰值性能为 55.296 TFLOPS。

1.2 层归一化

层归一化是一种用于神经网络训练的技术，它针对每个独立样本的特征向量进行归一化处理。相较于批量归一化，层归一化不依赖于小批量数据的统计特性，而是针对每个样本单独计算均值和方差，然后对数据进行归一化。这种处理方式有助于减少训练过程中网络隐藏层数据分布的变化，降低对参数初始化的敏感性，加快收敛速度，并提高网络的稳定性，在处理动态长序列等特定场景下，展现出了更加出色的性能。

层归一化的核心在于其标准化过程，该过程涉及对每个样本的所有特征进行均值和方差的计算，随后通过归一化、缩放和偏移 3 个步骤调整数据。

层归一化的计算如式（1）所示：

$$y=\frac{x-\mu}{\sqrt{\sigma^2+\varepsilon}}\times\gamma+\beta \tag{1}$$

其中：ε 是一个很小的常数，用于避免分母接近零时数值的不稳定；γ 是可训练的缩放参数；β 是可训练的偏移参数；μ 和 σ^2 是当前层输入样本特征向量的均值和方差。μ 的计算如式（2）所示：

$$\mu=\frac{1}{n}\sum_{i=1}^{n}x_i \tag{2}$$

其中：n 是层中隐藏神经元的个数。

σ^2 的计算方式和结果精度对于整个层归一化至关重要，因为它们会影响层归一化的计算效率和最终结果。对于 σ^2 的计算，主要有 Naive、Two Pass 和 Welford 三种不同的方法。

1）Naive 方法（直接计算法）：

Naive 方法的计算如式（3）所示：

$$\sigma^2=\frac{1}{n}\sum_{i=1}^{n}x_i^2-\left(\frac{1}{n}\sum_{i=1}^{n}x_i\right)^2 \tag{3}$$

该方法仅需遍历一次数据，可同时计算出数据的总和与平方和，进而通过数学运算求得 μ 和 σ^2，这在一定程度上提高了效率。但是，该方法在计算时会因累积舍入产生误差，对于大数据集而言，这种误差累积后致使结果 σ^2 无法满足精度要求；此外，该方法在处理大数据集时需存储数据的总和与平方和，因此会占用大量的内存资源。

2）Two Pass 方法（双遍法）：

Two Pass 方法根据方差的定义进行计算，μ 和 σ^2 的计算分别如式（2）和式（4）所示：

$$\sigma^2=\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2 \tag{4}$$

该方法以遍历两遍数据集的形式在数值稳定性和计算精度上优于 Naive 方法，但是这种形式也导致了计算成本的增加。由于需要对数据集进行两次遍历，该方法在处理大规模数据集时会消耗更多的计算时间和内存资源。由此产生的额外计算负担会成为制约高性能的关键因素，甚至可能导致处理速度大幅下降，进而形成性能瓶颈。

3）Welford 算法（单遍法）：

Welford 算法仅需遍历一次数据，同时能够在遍历过程中维护 μ 和 σ^2 两个变量。目前较多的深度学习框架都采用该算法，本文的代码采用的也是 Welford 算法。首先进行初始化，令 $\overline{x_1}=x_1$，$S_1=0$，随后每个数据点 $x_i(2\leqslant i\leqslant n)$ 的计算公式如式（5）～（7）所示：

$$\overline{x_i}=\overline{x_{i-1}}+\frac{x_i-\overline{x_{i-1}}}{i} \tag{5}$$

$$S_i=S_{i-1}+(x_i-\overline{x_{i-1}})(x_i-\overline{x_i}) \tag{6}$$

$$\sigma_n^2=\frac{S_n}{n} \tag{7}$$

其中：式（5）更新了当前数据点的均值 $\overline{x_i}$，$\overline{x_{i-1}}$ 是前 $i-1$ 个数据点的均值，x_i 是当前数据点；式（6）更新了中间变量 S_i，该变量可以逐步积累计算方差所需的数据；式（7）中的 S_n 是最后一个中间变量，用于计算最终结果 σ_n^2。

Welford 方法通过单次遍历数据集完成均值与方差的计算，缩短了计算周期，同时允许实时更新数据，无需将整个数据集保存于内存中，减少了内存占用。此外，该方法通过其迭代更新机制，有效抑制了累积误差，确保了计算结果的高精度与强稳定性。与 Naive 与 Two Pass 两种方法相比，在处理大规模数据或数据流时，该方法不仅提升了计算效率，同时也保障了数值稳定性。然而，Welford 方法存在一定局限性，一方面若初始值选取不当，计算结果的准确性会受初始均值与方差设定的影响，另一方面其实现过程相较于其他方法稍显复杂。

综上所述，在选择方案处理数据时，应综合考虑数据规模、应用场景需求、对计算精度的要求以及实现复杂度等多方面因素，科学合理地选用最适合的算法，来确保数据分析与处理的高效与准确。

2 层归一化的并行设计

针对层归一化中访存密集性的问题，本文提出了两种不同的层归一化并行设计方案：GPD 方案和 PDSS 方案。为了更明确地阐述分核方案，将 SW26010P 处理器上核组的个数定义为 N_p，每个核组上从核的个数为 N_c。

假定层归一化在计算机内存中的输入与输出数据均遵循 $NHWC$ 或 $NCHW$（N 代表批次中的图片数量，H 代表图片高度，W 代表图片宽度，C 代表输入通道数）的排列方式，其数据布局如图 2 所示。

式（1）中，x、γ 和 β 是输入参数，y、μ 和 σ^2 是输出参数，所有参数仅支持四维张量。x 和 y 的数据布局支持 $NHWC$ 和 $NCHW$ 两种排列方式，其数据布局方式需保持一致。γ 和 β 的数据布局通常是 λHWC 或

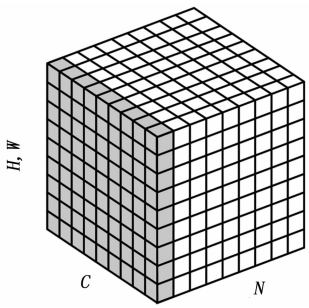

图 2　层归一化的数据布局

λCHW，其数据布局方式也需保持一致。μ 和 σ^2 数据布局是 $N\lambda\lambda\lambda$ 。其中，λ 为常数 1。$NHWC$ 和 $NCHW$ 的排列方式优化思路相同，所以下文的优化都以 $NHWC$ 为例展开。

要在从核上进行数据划分，首先需要将四维张量 $NHWC$ 处理成二维张量［N，HWC］再进行数据划分。其中，$HWC=H\times W\times C$。

2.1　GPD 方案

基于 SW26010P 众核处理器任意场景下的 GPD 方案的分核策略和张量数据划分主要分为以下两个部分：

1）对单个核组的从核阵列：将每个一维向量 HWC 分配到一个从核上，也就是对二维张量的第一个维度 N 进行数据划分。具体来说，以从核数量 N_c 作为一轮计算的标准，第 $n(n\leqslant N_c)$ 个从核上可分配到 i_n 个一维向量，经过 j 轮完成数据分配。i_n 和 j 的计算公式如式（8）和式（9）所示：

$$i_n=\begin{cases}\left\lceil\dfrac{N}{Nc}\right\rceil & n<(N \bmod N_c)\\ \left\lfloor\dfrac{N}{N_C}\right\rfloor & \text{其它}\end{cases} \tag{8}$$

$$j=\left\lceil\frac{N}{Nc}\right\rceil \qquad N\geqslant 1 \tag{9}$$

其中：“⌈ ⌉”代表向上取整，“⌊ ⌋”代表向下取整。

2）针对从核私有存储的限制，本文对二维张量的第二个维度 HWC 进行了划分：由于每个从核仅有 256 kB的私有存储空间，需要将输入数据一维向量 HWC 进一步切割成 m 个子块，m 的计算公式为：

$$m=\begin{cases}M & HWC>M\\ 1 & 1\leqslant HWC\leqslant M\end{cases} \tag{10}$$

子块上的输入数据 x、γ 和 β 以及计算结果的输出数据 y、μ 和 σ^2 所占的存储空间不应该超过 256 kB，所以式（10）中 M 的计算公式为：

$$M=\frac{256\text{kB}-(X_{\text{buffer}}+\gamma_{\text{buffer}}+\beta_{\text{buffer}}+Y_{\text{buffer}})}{HWC\times 1} \tag{11}$$

其中：X_{buffer}、γ_{buffer} 和 β_{buffer} 为输入数据所需的存储空间，Y_{buffer} 为缓冲输出数据所需的存储空间。

层归一化满足任意场景的 GPD 方案框架如图 3 所示。首先，每个从核需要加载输入数据 x、γ 和 β，这些数据通过 DMA 机制，依据预设的索引加载到 LDM 上的指定空间；其次，从核获取到输入数据后进行层归一化的计算；最后，在完成正向计算后，从核需要保存 μ 和 σ^2 用于反向传播的计算，同时，将输出数据 y 写回主存。

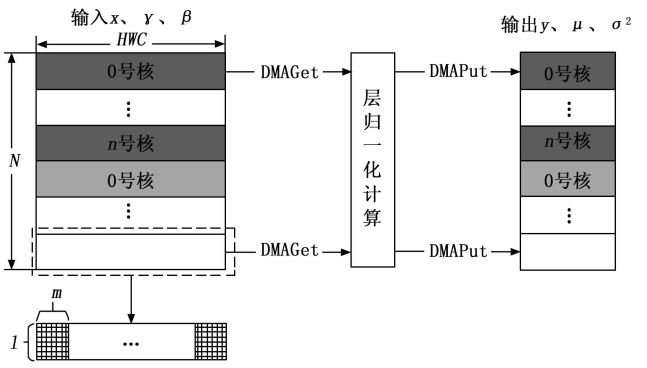

图 3　GPD 框架图

该并行设计方案虽确保了跨场景的广泛适用性，但其泛化性往往难以保证在所有数据场景下均实现高性能。这是因为在不同的应用场景中，数据的分布、特征的维度以及计算资源的可用性等因素存在显著差异，这些因素可能会影响并行设计方案的效率和性能表现。因此，针对上述问题，本文在第 2.2 节中针对部分场景提出了不同于 GPD 方案的并行优化方案。

2.2　PDSS 方案

基于 SW26010P 众核处理器的 PDSS 方案的分核策略和张量数据划分分为以下两个部分：

1）针对单个核组的从核阵列的任务划分：对二维张量的第二个维度 HWC 先进行划分。将 HWC 划分为 N_c 个 HWC_{blk}（block of HWC），HWC_{blk} 的计算公式为：

$$HWC_{\text{blk}}=\left\lfloor\frac{HWC}{N_c}\right\rfloor+[n<(HWC \bmod N_c)] \tag{12}$$

2）从核私有存储的数据切割：将输入数据一维向量 N 进一步切割成 m 个子块，其中，$m=N$。由于每个从核仅有 256 kB 的私有存储限制，所以子块上的输入张量 x、γ 和 β 以及计算结果张量 y、μ 和 σ^2 所占的存储空间不应该超过 256 kB，由此得出 HWC_{blk} 需满足以下关系式才可进入该场景下的优化：

$$HWC_{\text{blk}}<256\text{ KB}-(X_{\text{buffer}}+\gamma_{\text{buffer}}+\beta_{\text{buffer}}+Y_{\text{buffer}}+\mu_{\text{buffer}}+\sigma^2_{\text{buffer}}) \tag{13}$$

其中：X_{buffer}、γ_{buffer} 和 β_{buffer} 为输入数据所需的存储空间，Y_{buffer}、μ_{buffer} 和 σ^2_{buffer} 为缓冲输出数据所需的存储空间。

层归一化 PDSS 方案框架如图 4 所示。该方案数据的加载和写回过程与 2.1 节数据的加载和写回过程

相同。

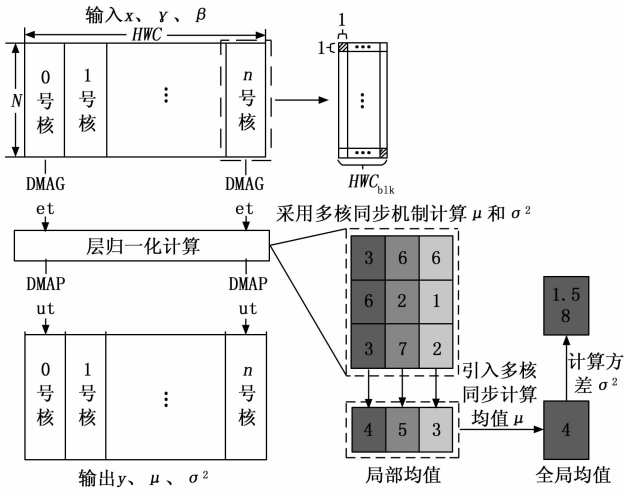

图 4 PDSS 框架图

PDSS 方案的划分方式致使各个数据块的 μ 与 σ^2 分布在 N_c 个从核上，为确保计算出整体数据的最终 μ 与 σ^2，需引入多核同步机制实现数据的准确计算。以形状 $N=3$、$HWC=3$ 的二维张量为例：首先，对于每个从核所负责的数据块沿着 N 维度计算局部均值。然后，利用多核同步机制，将这些局部均值聚合起来，计算出整体数据的全局均值。最后，基于全局均值，进一步计算出整体数据的方差。该机制并非依赖于传统的软件同步方法，而是直接利用了硬件封装好的接口，其底层原理基于片上同步网络（On-Chip Synchronization Network）。通过这种硬件接口，多核之间的同步操作能够以极高的效率和安全性完成，避免了传统软件锁机制所带来的开销和性能瓶颈。

3 归一化的并行优化方案

3.1 双缓冲优化

在串行数据处理方案中，从核通过 DMA 技术从主核拉取数据至本 LDM 空间，随后执行层归一化计算任务，最终将结果写回主核。此过程中，在数据获取和写回时，从核处于闲置状态，导致计算资源未充分利用，在一定程度上影响了层归一化的性能。为解决这一问题，本文引入双缓冲优化手段，即并行化数据处理与通信过程来增强系统吞吐率和响应速度。

如图 5 所示，本文将内存设为两个独立的缓冲区（Buffer 0 与 Buffer 1），其容量大小可以依据实际应用需求与数据流特征定制。这种设计的核心在于实现计算与通信的并行化，有效隐藏了各自的开销。交替使用这两个缓冲区，确保一个缓冲区在读取和写回数据时，另一个缓冲区用于计算任务，从而实现计算与数据通信的并行处理。双缓冲技术使得除首尾子块外，其余子块的计算与通信得以重叠，减少了时间开销，从而提高层归一化效率。

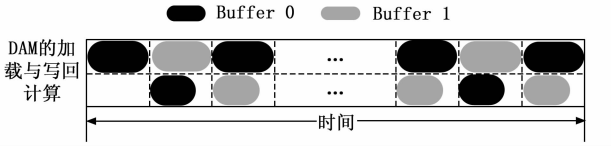

图 5 双缓冲示意图

在 GPD 方案中，双缓冲机制主要体现在每个从核内部的 m 个子块之间，通过在相邻子块之间交替加载和处理数据，实现了高效的流水线操作，从而减少了数据加载和计算之间的等待时间。而在 PDSS 方案中，双缓冲机制则在两个相邻从核之间 N 个 HWC_{blk} 大小的子块，通过在不同从核的子块之间进行数据的交替传输和处理，不仅优化了单个从核内的数据流，还增强了从核之间的协作效率，进一步提升了整体的并行处理能力。这种设计充分利用了众核架构的并行性，有效缓解了数据传输瓶颈，显著提高了系统的吞吐量和性能表现。

对于多轮 DMA 操作的应用场景，需在 LDM 中分配两倍于单次通信数据量的空间，用以存放互为备份的数据集，确保计算的连续性与高效性。DMA 技术和双缓冲相结合的方式操作流程如下：

1）初始时，DMA 将首批数据加载至 Buffer0。

2）从核开始处理 Buffer0 数据并进行计算，随后将计算结果写回 Buffer0，同时 DMA 加载下一批数据至 Buffer1。

3）循环第一步和第二步的操作过程，直至所有从核完成计算任务。

4）DMA 将最后一批计算结果写回主存，完成整个流程。

3.2 SIMD 优化

SIMD 指令[20]允许在同一指令周期内对一组数据执行统一的操作，是一种高效的并行处理技术。其工作原理是将多个数据打包成一个数据向量，随后通过单条指令对这一向量内的所有数据执行相同的运算，通过减少指令数降低对访问带宽的要求，提升了数据处理的吞吐量与效率。这一技术的实现通常需要向量处理器或是 CPU 内置的向量寄存器。SIMD 的核心优势主要有高效数据吞吐和并行计算加速两方面：针对连续内存数据，单条向量加载/存储指令即可完成批量数据传输，减少指令开销，向量寄存器内的数据可同步执行运算操作，实现单指令多数据并行处理。

在 SW26010P 处理器中，主核支持的 SIMD 指令处理长度为 256 位，从核支持的 SIMD 指令处理长度为 512 位。执行层归一化计算任务时，在从核上通过 SIMD 指令进行向量化处理，使用了数据类型为 floatv8 的向量计算，该类型的 SIMD 变量可以存储 8 个单精度

浮点类型的数据，需注意 floatv8 类型是 32 字节对界。

以 PDSS 优化方案中 SIMD 指令计算 μ 和 σ^2 为例：首先，通过 SIMD 指令对数据进行并行加载和计算，这充分利用了 SIMD 指令集的特性，能够同时处理多个数据元素，从而提高数据加载的效率。随后，利用 SIMD 指令集的逐元素操作功能，高效地执行向量化的加法和乘法运算。这些运算不仅能够快速完成对数据的初步处理，还能在单个指令周期内对多个数据点进行计算，从而提高了计算效率。同时通过这种方式，从核迅速地计算出每个数据块的局部 μ 和 σ^2，为后续的全局统计量计算提供了必要的中间结果。最后，使用 SIMD 指令集对计算结果进行并行存储，并将这些局部统计量有效聚合起来，以计算出全局均值和方差。

4　实验评估

4.1　测试环境

本文的测试是基于 SW26010P 处理器进行性能评估，其具备 51.2 GB/s 的访存带宽，256 kB 从核 Cache 容量和 16 GB 的总内存容量，设置核组的数量 $N_p=1$，每个核组上从核的数量 $N_c=32$。为了全面覆盖并验证优化场景的有效性，选取了 6 个具有代表性的测试样例进行测试，见表 1。

表 1　测试样例

测试样例	输入数据 x	输入数据 γ / β	输出数据 y	输出数据 μ / σ^2
1	25×31×301×1	1×31×301×1	25×31×301×1	25×1×1×1
2	2 048×1×1×12 288	1×1×1×12 288	2 048×1×1×12 288	2 048×1×1×1
3	78×3×3×1 649	1×3×3×1 649	78×3×3×1 649	78×1×1×1
4	800×4×1×748	1×4×1×748	800×4×1×748	800×1×1×1
5	2 048×2×2×1 024	1×2×2×1 024	2 048×2×2×1 024	2 048×1×1×1
6	866×5×1×512	1×5×1×512	866×5×1×512	866×1×1×1

为全面分析该架构下层归一化的效率，采用的 3 种测试方法均基于 C 语言编程与 Athread 线程库实现，以 NVIDIA 提出的 cuDNN 计算库中标准层归一化计算结果作为正确性基准，确保实验环境的统一性和可比性。其中，这 3 种方法包括单核上的串行层归一化、多核上的并行层归一化以及通过 SIMD 指令进行数据并行优化的并行层归一化。所有测试都使用单精度浮点计算，SIMD 指令使用的数据类型为 floatv8。为保证结果的稳定性，取算法运行 10 次的平均运行时间作为测试结果。性能测试指标为加速比 S_G，用于衡量并行系统或算法相对于串行执行时的性能提升，其定义为：

$$S_G=\frac{T_{\text{serial}}}{T_{\text{parallel}}} \tag{14}$$

其中：T_{serial} 表示在主核串行执行算法所需的时间，T_{parallel} 表示在从核优化后并行执行算法所需的时间。

4.2　实验结果分析

4.2.1　精度损失结果与分析

如表 2 所示，测试样例中输出数据 y 的最大绝对误差均不超过 16.26×10^{-2}，最大平均绝对误差为 6.11×10^{-6}。这充分说明第四章提出的层归一化加速计算技术得到的计算结果与使用 NVIDIA 提出的 cuDNN 计算库中标准层归一化计算结果相近，结果具有可靠性。

表 2　测试样例精度结果

测试样例	平均绝对误差($\times10^{-7}$)	最大绝对误差($\times10^{-2}$)
1	4.08	0.64
2	4.36	16.26
3	4.82	3.42
4	4.15	2.24
5	4.15	6.07
6	6.11	8.23

4.2.2　并行层归一化算法加速效果分析

图 6 展示了在单个核组上，本文通过实施双缓冲策略优化后的并行归一化计算所取得的加速效果，与之对应每个测试样例的运行时间如图 7 所示。实验结果表明，经过双缓冲优化的并行化 GPD 方案能够大幅缩短处理时间，其中最高加速比达到了 55.48，平均加速比为 36.60，充分体现了双缓冲机制在提升计算效率方面的强大优势。

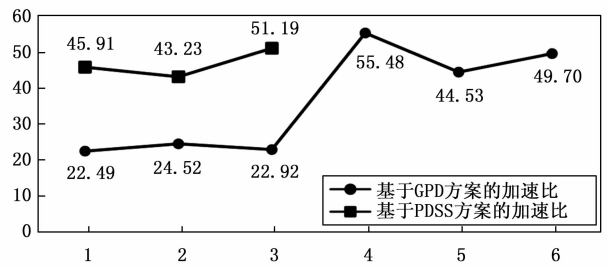

图 6　测试样例的主从加速比

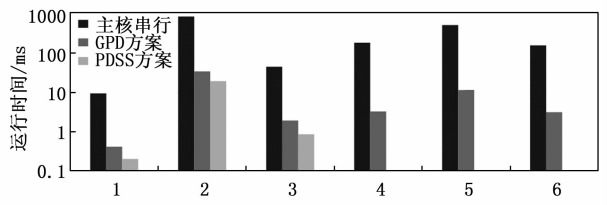

图 7　测试样例在主核串行和从核并行上的运行时间对比

然而，尽管层归一化的整体性能提升显著，测试样例 1、2、3 的优化效果并未达到与其他样例同等的加速

水平。究其原因，层归一化操作本质上具有较高的访存密集性，在一定的数据规模范围内，尽管各从核上的计算处理能够迅速完成，但数据的加载与写回操作却耗时较长。这一瓶颈限制了计算与访存操作之间的有效重叠，进而削弱了加速性能，导致双缓冲优化的优势未能在所有测试样例中得以充分体现。这一现象表明，尽管双缓冲策略在提升并行计算效率方面发挥了重要作用，但在面对具有高访存需求的层归一化任务时，仍需进一步优化数据传输机制，以实现计算与访存操作的更高效协同，从而进一步提升整体优化效果。

鉴于此，通过对特定场景实施 PDSS 方案优化，测试样例 1 的加速比从 22.49 增长至 45.91，测试样例 2 的加速比从 24.52 增长至 43.23，测试样例 2 的加速比从 22.92 增长至 51.19。这表明，本文提出的 PDSS 方案对一定场景的优化有效提升了并行处理的效率和加速效果。

4.2.3 优化并行层归一化算法及优化效果分析

图 8 和图 9 分别从运行时间和有效算力两个维度，展示了在双缓冲机制下，通过应用 SIMD 指令集优化，对并行数据处理实现的加速效果。

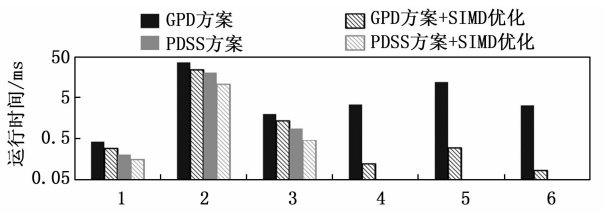

图 8 使用 SIMD 指令优化后的运行时间的对比

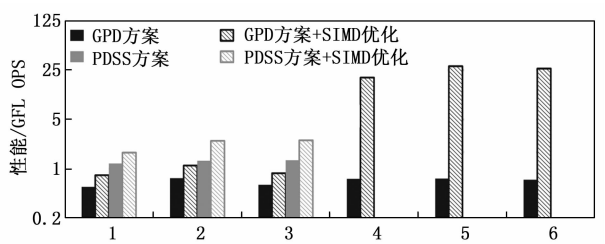

图 9 使用 SIMD 指令优化后的性能对比

实验结果表明，经过 SIMD 指令集优化后，每个测试样例的运行时间显著降低，有效算力大幅增加。其中，最大有效算力可达到 28.25 GFLOPS。由此可以看出，在优化并行计算方面，数据间的并行处理是一种有效的优化方法。同时，对测试样例 1、2 和 3 的性能进行分析发现，尽管 GPD 方案采用了 SIMD 优化，其性能表现仍未达到 PDSS 方案的性能。这一结果表明，虽然 SIMD 优化能够有效实现数据的并行处理并显著提升计算速度，但在处理具有高访存需求的层归一化任务时，数据传输和分核策略的优化同样至关重要。因此，针对不同应用场景的特点，制定差异化的优化策略是实现性能提升的关键所在。

在处理测试样例时，计算过程中由于数据尾块大小不足 32，引入了额外的时间开销，从而使部分测试样例加速效果未能达到理论上的最优水平。但相较于未经 SIMD 指令集优化的并行层归一化算法，优化后的算法在算力提升上取得了显著成效。

5 结束语

本文从解决层归一化在高性能中具有访存密集性这一问题的角度出发，设计了两种不同的优化分核策略，通过双缓冲机制、DMA 技术和 SIMD 向量化等优化手段，实现了层归一化的异构并行化处理。实验结果表明，在单个核组下，本文提出的并行层归一化在 GPD 方案中最高加速比可达 55.48，平均加速比可达 36.60，在 PDSS 方案中最高加速比可达 51.19。除此之外，使用了 SIMD 指令的并行优化设计后，最大有效算力可达 28.25 GFLOPS。层归一化所存在的访存密集性问题，在面对不同的数据场景以及多样化的数据类型时，依然有待进一步优化完善。未来的研究重点在于，当将计算任务细化并分配到各个从核以维持高度并行性的过程中，怎样才能最大程度地削减核间通信所产生的开销，以及如何实现 SIMD 指令使用的最优化，也是关键所在。

参考文献：

[1] LECUN Y，BENGIO Y，HINTON G. Deep learning [J]. Nature，2015，521 (7553)：436－444.

[2] LIU Y，GAO Q，YANG Z，et al. Learning with adaptive neighbors for image clustering [C] //International Joint Conference on Artificial Intelligence，2018：2483－2489.

[3] GIRSHICK R，DONAHUE J，DARRELL T，et al. Rich feature hierarchies for accurate object detection and semantic segmentation [C] //Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition，2014：580－587.

[4] NOH H，HONG S，HAN B. Learning deconvolution network for semantic segmentation [C] //Proceedings of the IEEE International Conference on Computer Vision，2015：1520－1528.

[5] HUANG G，LIU Z，LAURENS V，et al. Densely connected convolutional networks [C] //IEEE Computer Society，2016：1－8.

[6] LOFFE S，SZEGEDY C. Batch normalization：accelerating deep network training by reducing internal covariate shift [C] //International Conference on Machine Learning，2015：448－456.

[7] DOROBANTU V，STROMHAUG P A，RENTERIA J. Dizzymn：reparameterizing recurrent neural networks for

norm-preserving backpropagation [J]. Arxiv Preprint Arxiv: 1612.04035, 2016.
[8] ARJOVSKY M, SHAH A, BENGIO Y. Unitary evolution recurrent neural networks [C] //International Conference on Machine Learning, 2016: 1120 - 1128.
[9] BA J L, KIROS J R, HINTON G E. Layer normalization [J]. Arxiv Preprint Arxiv: 1607.06450, 2016.
[10] PARMAR N, VASWANI A, USZKOREIT J, et al. Image transformer [C] //International Conference on Machine Learning, 2018: 4055 - 4064.
[11] VASWANI A, SHAZEER N, PARMAR N, et al. Attention is all you need [C] //Neural Information Processing Systems, 2017: 5998 - 6008.
[12] ZHOU S, DONG L, XU S, et al. Syllable-based sequence-to-sequence speech recognition with the transformer in mandarin Chinese [C] //Conference of the International Speech Communication Association, 2018: 791 - 795.
[13] DAGA M, AJI A M, FENG W C. On the efficacy of a fused CPU + GPU processor (or APU) for parallel computing [C] //2011 Symposium on Application Accelerators in High-Performance Computing, Knoxville, 2011: 1 - 6.
[14] KECKLER S W, DALLY W J, KHAILANY B. GPUs and the future of parallel computing [J]. IEEE Micro, 2011, 31: 7 - 17.
[15] CARTER N P, AGRAWAL A, BORKAR S. Runnemede: an architecture for ubiquitous high-performance computing [C] //2013 IEEE 19th International Symposium on High Performance Computer Architecture, Shenzhen, IEEE, 2013: 1 - 10.
[16] FU H, LIAO J, YANG J, et al. The sunway taihu light supercomputer: system and applications [J]. Science China Information Sciences, 2016, 59 (7): 1 - 16.
[17] 高　捷，刘　沙，黄则强，等. 基于国产众核处理器的深度神经网络算子加速库优化 [J]. 计算机科学，2022，49 (5)：355 - 362.
[18] FU H, HE C, CHEN B, et al. 18.9-PFLOPS nonlinear earth-quake simulation on sunway Taihu light: enabling depiction of 18 Hz and 8-meter scenarios [C] //Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2017: 1 - 12.
[19] XU Z, LIN J, MATSUOKA S. Benchmarking sw26010 many-core processor [C] //2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), IEEE, 2017: 743 - 752.
[20] 莫尚丰，周振芬，胡勇华，等. 基于 FT-M7002 的复数域行向量矩阵乘法移植与优化 [J]. 计算机科学，2023，50 (s2)：839 - 844.

(上接第 173 页)

[7] 姜艳娜，许彦章，李媛媛，等. 一种弹载设备快速软件在线升级设计与实现 [J]. 电子技术应用，2023，49 (6)：109 - 113.
[8] 万　垚，李　鑫. 一种基于 FPGA 的在线升级方案 [J]. 成都信息工程大学学报，2020，35 (5)：493 - 498.
[9] 韩子舟，任勇峰，李辉景. 基于 1553B 总线的 FPGA 在线升级 [J]. 电子设计工程，2022，30 (3)：1 - 5.
[10] 李　伟，宋　燕，龙　燕. 一种基于 SPI 配置模式 FPGA 在线升级方案 [J]. 电子世界，2019 (6)：37 - 38.
[11] 赵冬青，梁　璠，上官鹏，等. 一种基于 SPI FLASH 的 FPGA 固件更新方法 [J]. 电子设计工程，2020，28 (16)：11 - 16.
[12] 张永乐，王永勇，郑　炜. 一种基于 FPGA 的在线程序升级方案 [J]. 电子技术应用，2017，43 (3)：48 - 50.
[13] 林天静，阮　翔，刘　春. 基于 FLASH 控制器的 FPGA 在线加载功能设计 [J]. 电子技术应用，2019，45 (1)：88 - 91.
[14] 王　锋，吕天志，杨明洋. 一种基于 PCIe 总线的 SPI-Flash 内 FPGA 程序在线更新方法 [J]. 电子制作，2021 (9)：56 - 59.
[15] 严紫薇，陈少华，张宝朋. 采用 RS 编码的改进高速数据传输平台实现 [J]. 单片机与嵌入式系统应用，2021，21 (5)：79 - 83.
[16] 汪智杰，周治柱. 基于国产 FPGA 的 RS 编译码器设计与应用 [J]. 电子设计工程，2022，30 (22)：140 - 144.
[17] CANZE Z, QUNYING L. The [1, 0] -twisted generalized Reed-Solomon code [J]. Cryptography and Communications, 2024, 16 (4): 857 - 878.
[18] PEJMAN M, JENS V, DRIES V, et al. Resilience of reed-Solomon codes against single-frequency electromagnetic disturbances: fault mechanisms and fault elimination through symbol inversion [J]. Electronics, 2022, 11 (9): 1292 - 1292.
[19] WAHBIA, IDRISSI E H E A, ROUKHEA, et al. Design, optimization and real time implementation of a new embedded Chien search block for reed-Solomon (RS) and Bose-Chaudhuri-Hocquenghem (BCH) codes on FPGA board [J]. International Journal of Communication Networks and Information Security, 2021, 13 (1): 9 - 14.
[20] ZHANG D, ZHANG Z, ZHENG F. Iterative decoding algorithm for LDPC-RS product codes based on joint check [J]. Journal of Physics: Conference Series, 2024, 2849 (1): 012104.