论文阅读记录

1. 标题

SCALEDEEP: A Scalable Compute Architecture for Learning and Evaluating Deep Networks

1. 发表会议/期刊（年份）

ISCA.2017.

1. 作者及单位

Swagath Venkataramani, School of ECE, Purdue University

1. 所提出方法的名称

SCALEDEEP（Scalable Compute Architecture for Learning and Evaluating Deep）

1. 所提出方法的流程图

背景和问题

DNN在涉及自然语言、语音、图像和视频处理的广泛任务上的显示出最先进的性能，并部署在许多现实世界应用程序中。然而，由于网络的复杂性和它们处理的数据量，**DNNs带来了巨大的计算挑战（嵌入式推理和基于云的训练）**，预计这两者在未来都将增长。

思想

为了提高DNNs的效率，提出了**SCALEDEEP，这是一种密集的、可伸缩的服务器架构，它的处理、内存和互连子系统都是专门用来利用DNNs的计算和通信特性的**。关键的区别在于SCALEDEEP**主要针对DNN训练**，而不是仅仅进行推理或评估。

SCALEDEEP的**主要架构特征**是:(1)异构处理分片和芯片以匹配广泛多样性的计算特征 (FLOPs and Bytes/FLOP ratio),其显示不同的粒度级别,(2)内存层次结构和三层网轮环grid-wheel-ring互连拓扑适合DNN的内存访问和通信模式，它匹配DNNs的一层内的关键内存访问和通信模式，跨层的生产者-消费者关系，以及在一个小批处理中跨输入的数据与模型并行性,(3)基于硬件数据流跟踪的低开销同步机制,(4)将DNNs映射到所提出的体系结构的方法，通过嵌套流水线最小化数据移动，提高核利用率。

开发了一个编译器，允许将任何DNN拓扑编程到SCALEDEEP上，并开发了一个详细的架构模拟器来评估性能和能耗。

性能与功耗

该模拟器结合了基于英特尔14nm技术综合的SCALEDEEP s组件的时序和功率模型。评估了SCALEDEEP的一个实施例，其使用7032个处理块，工作在600MHz，峰值性能为680 TFLOPs(单精度)和1.35 PFLOPs(半精度)，总功耗为1.4KW。通过包含0.65M-14.9M神经元和6.8M-145.9M权重的11个最先进的DNNs，包括来自5年ImageNet比赛的获胜者，SCALEDEEP比最新GPU实现性能展示了6×-28×恒功率加速。

每个ConvLayer和FcLayer芯片分别消耗∼58W和∼15W的功耗。这将导致芯片集群级的能耗约为325W（4个ConvLayer和1个FcLayer），完整的SCALEDEEP节点为1.4KW（4个芯片集群）。SCALEDEEP的最高处理效率为485.7 GFLOPs/W，平均331.7 GFLOPs/W，而单个CompHeavy的最高处理效率可达934.6 GFLOPs/W。

应用场景

DNN训练和推理加速。

优缺点

优点在于SCALEDEEPSCALEDEEP是一种全系统(服务器节点)体系结构，关注DNN训练的挑战。

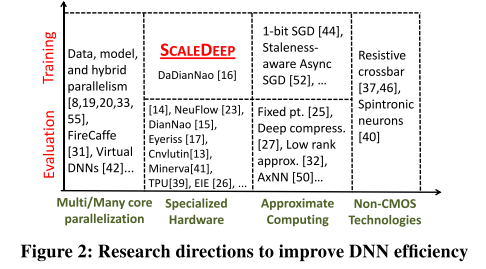
平台

用c++开发了一个详细的、周期精确的SCALEDEEP架构模拟器。模拟器严格地对每个执行周期中发生的所有事件建模，包括计算操作、片上/片外内存访问和数据传输。

为了估计计算能力，在RTL中实现了CompHeavy和MemHeavy tile执行阵列，将它们综合到Intel 14nm技术节点，并使用Synopsys设计编译器在门级测量了计算能力。使用为14nm节点开发的内部模型估计片内/片外存储器和互连电源。将各部件消耗的功率并入模拟器，利用仿真过程中观察到的执行轨迹估计能量。

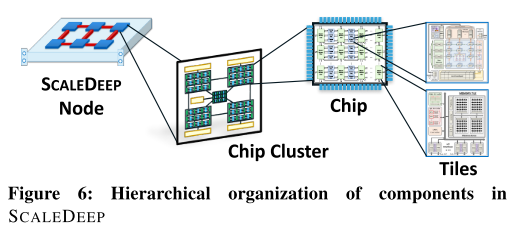
细节

相关工作



**SCALEDEEP架构**

图6显示了SCALEDEEP的层次结构。自底向上，SCALEDEEP由两种类型的处理块组成，每一种处理块都包含处理元素数组和scratchpad。在下一层，将处理块连接起来形成SCALEDEEP芯片。多个芯片被组织成芯片集群，在最顶层，几个芯片集群一起形成SCALEDEEP节点。。



**3.1异构处理块**

SCALEDEEP积极地**利用了层内和层间计算的异质性**。将深度网络中的计算划分为两组:(i)以计算为主并为数据重用(低Bytes/FLOP)提供充足机会的操作，即nD-卷积和矩阵乘法。(ii)操作简单，内存占主导地位(高Bytes/FLOP 比)，即nD-累加、子/上采样、激活函数和向量元素乘。因此，设计了两种类型的处理块—**Compute-Heavy (CompHeavy)块和****Memory-Heavy (MemHeavy)块**—来分别实现计算和内存占主导地位的操作。

**Compute-Heavy Tile**

它由可重构的二维处理元素数组(2D-PEs)组成。每个2D-PE都包含一个融合了乘法和累加单元的向量。一个1D累加器阵列位于2D阵列的右边界。沿着左侧、顶部和底部边框放置了三组流内存(SM)元素，使用它们将操作数提供给2D-PEs。它还包含一个小的scratchpad来保存来自2D数组的临时输出。CompHeavy块中的其他组件包括指令内存、解码和控制单元、标量寄存器文件和顺序标量处理元素，用于执行控制操作，如循环测试、指针算术、分支等。

优化后的CompHeavy tile可以进行批量卷积(一个输入，多个内核)和矩阵乘法运算。例如，批量2D卷积实现如下。卷积输入的行沿着二维数组的行进行馈电，而核(权矩阵)行沿着列进行馈电。每个2D-PE计算输入行与内核行之间的点积。卷积输出是通过对1D累加器阵列中的点积进行对角累加得到的。对于某些卷积输出，并不是所有行方向的点积都是在2D数组的相同执行迭代中产生的。在这种情况下，部分输出存储在本地scratchpad中，当剩余的点积可用时，由1D累加器数组取回。由于2D-PE有多个执行通道，因此可以同时提供相同数量的内核，从而允许并行计算同一输入上的多个卷积。

**阵列可配置性**

允许一些微架构参数在运行时配置。2D数组的列和向量通道可以重新分布，同时保持它们的乘积为常数。2D数组可以水平分割为两个2D数组，每个数组的行数都是原来的一半。左边的SMs也被一分为二，而顶部和底部的SMs专门提供给较小的2D数组的列。

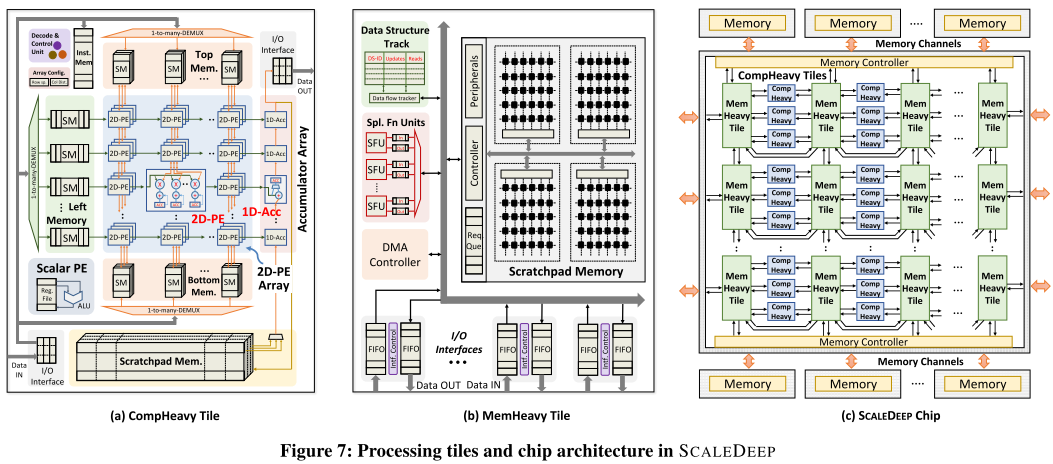
**Memory-Heavy**

包含一个大型的scratchpad内存，存储特征、权重、误差和误差梯度，以及直接与之连接的一组特殊函数单元(SFUs)。SFUs包含一个加法器/比较器、一个乘法器和支持激活函数(如ReLU、tanh和sigmoid)所需的逻辑。在MemHeavy块中执行nD-累加、激活函数、子/上采样和向量元素乘等操作。MemHeavy块包括一个DMA控制器，用于在内存阵列之间进行数据传输。此外，MemHeavy块还包括硬件数据流跟踪器，用于跟踪对选定地址范围的读写访问。由于SCALEDEEP不包含硬件管理的缓存或一致性机制，所以使用这些跟踪器在处理块之间同步。

**3.2芯片架构**

**架构描述**

处理块被布置为一个二维网格，由CompHeavy块和MemHeavy块交替组成列。该体系结构包含每MemHeavy块3个CompHeavy块，每个块对应FP、BP和WG。每个CompHeavy块都连接到其左右的MemHeavy块。此外，每个MemHeavy 块都直接连接到上下左右MemHeavy 块。外部存储器连接到SCALEDEEP芯片的顶部和底部边界。SCALEDEEP芯片中的所有互连都是点对点连接，没有仲裁。



**执行模型和SCALEDEEP ISA**

每个CompHeavy块运行一个执行线程，其程序存储在指令内存中。SCALEDEEP中的内存层次结构完全由软件管理，没有传统的缓存、一致性或同步机制。ISA包含28条指令，图8中列出了其中的一个子集。这些指令可分为5类：

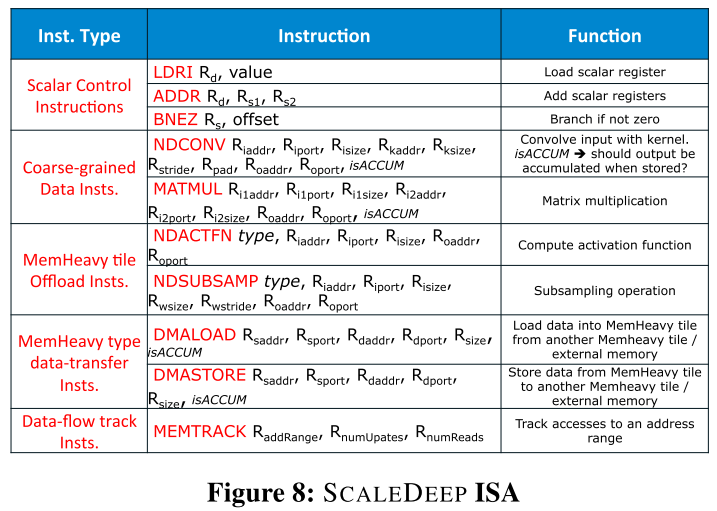
1）标量控制指令:用于确定程序控制流的加载/存储、标量操作和分支指令。它们在CompHeavy 块的标量PE上执行。

2）粗粒度数据指令:计算主要指令，如卷积(NDCONV)等，在CompHeavy块的PE数组上执行。

3）MemHeavy块卸载指令:高Bytes/FLOP指令，如向下采样(NDSUBSAMP)等，卸载到一个连接的MemHeavy块。

4）MemHeavy数据传输指令:启动连接的MemHeavy数据传输的指令。

5）数据流跟踪指令:用于跟踪对指定数据结构或地址范围的访问的指令。它们用于强制同步执行。

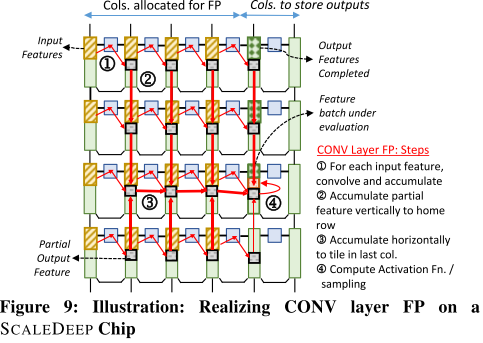


**在芯片上实现DNN层**

根据计算和内存需求为每一层分配一组列，DNN层在空间上分布在整个芯片上。

**分布式网络状态和本地化计算**：

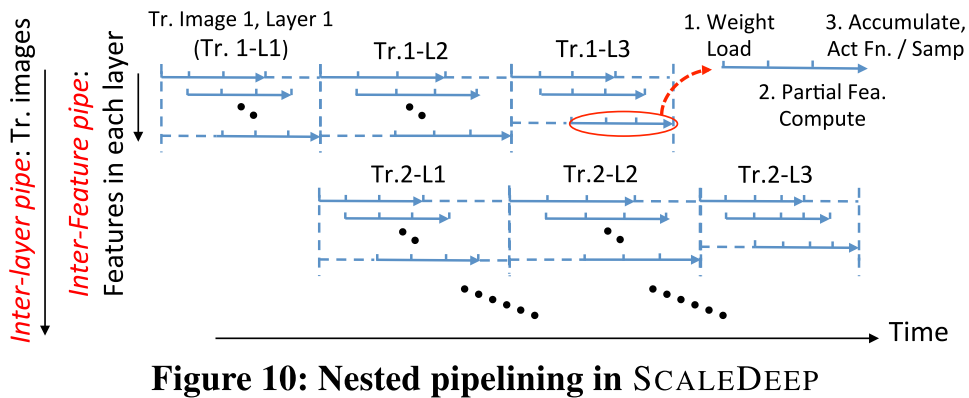
网络中的每个特征和误差数组都被分配一个home MemHeavy tiles。所选层的权重和权重梯度存储在片上，存储在包含相应特征存储的MemHeavy tiles中。其他层的权重和权重梯度存在外部存储，并预取进MemHeavy。CompHeavy tiles生成并使用存储在直接连接到它们的MemHeavy tiles中的网络状态。MemHeavy tiles中的SFUs也对存储在其中的网络状态进行操作。因此，通过在SCALEDEEP芯片的许多处理块上对网络状态和相关计算进行空间分区，可以本地化数据移动并最小化互连带宽。这一点，加上一个简化的内存层次结构，大大提高了SCALEDEEP的能源效率。



**实现层序列：**

DNN的连续层被映射到SCALEDEEP芯片中相邻的列集。这利用了DNN层之间的生产者-消费者关系。BP和WG的实现方式相似，只是它们使用各自的处理块集，并且数据流的方向相反。权值梯度可以存储在外部存储器中，也可以存储在芯片上，当该层的BP/WG完成后，就可以丢弃误差。

**内嵌流水线**：芯片编程为如同一个2级内嵌流水线操作。在最内层是特征间流水线，其中计算评估输出特征批(即可选权重加载、部分特征输出评估和特征积累/激活函数/采样)形成流水线。在最外层是层间流水线，对应于每一层的块利用小批量并行性以流水线的方式在连续的训练/评估输入上操作。



**利用数据流跟踪器同步**

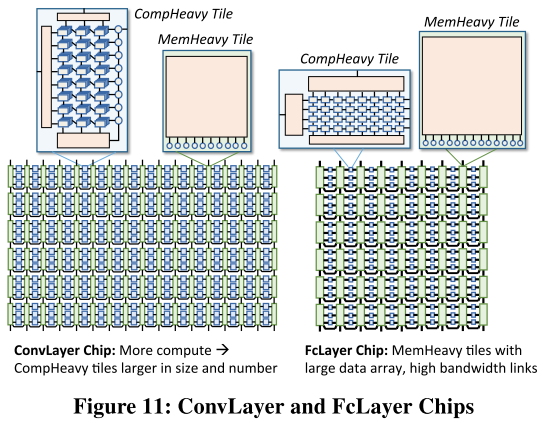
观察到数据访问在编译时可确定且累加是可交换的，新的同步原语：

MEMTRACK(AddRange,NumUpdates,NumReads)

CompHeavy tile将MEMTRACK指令卸载到适当的MEMTRACK tile，然后使用硬件计数器跟踪对地址范围的访问，并确保访问顺序符合规范。通过确保对内存位置的访问顺序遵循指定的模式来强制同步。

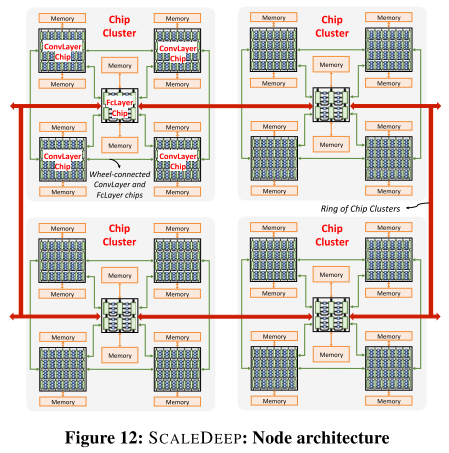
**异构芯片设计**

使用架构模板构建了两个不同的芯片，即ConvLayer芯片和FcLayer芯片。针对2D阵列的行数量、列数量、通道数量、内存组织以及片上片外连接进行微调。



**3.3 节点架构**

多个ConvLayer和FcLayer芯片连接在一个两层的层次结构中。



**Chip Cluster - Wheel of ConvLayer andFcLayer Chips**

节点级的一个关键挑战是FcLayer芯片需要更高的内存带宽来保持与ConvLayer芯片相同的吞吐量。减少内存带宽的一种方法是将输入聚合到FC层，并在FcLayer芯片中作为批处理执行它们。这允许每批只获取一次层参数，从而降低了与批大小成比例的带宽。因此，如图12所示，将多个ConvLayer芯片和一个FcLayer芯片连接成一个轮子，形成芯片集群。ConvLayer位于圆周上，而FcLayer位于中心。ConvLayer芯片并行地操作不同的网络输入，而FcLayer芯片接收来自所有ConvLayer芯片的FC层输入，并对其进行批处理。平衡了FcLayer芯片的计算能力，使其与所有连接到它的ConvLayer芯片的吞吐量相匹配。这限制了车轮上的ConvLayer芯片的数量。

轮形结构降低了外部内存容量，因为FC层的权重(占总权重存储需求的主导)只需要存储在连接到FcLayer芯片的内存中。此外，请注意，车轮的弧连接到每个ConvLayer芯片与相邻的芯片。这使得CONV层可以跨多个芯片进行分区，以防它们不能安装在单个芯片上。然而，这样做减少了FcLayer芯片的批量大小，因为只有一些轮弧被利用。轮弧还用于累积权重梯度，并在每个小批量后分配更新后的权重。

**Ring of Chip Clusters**

为了进一步利用并行性，将多个芯片集群通过它们的FcLayer芯片连接到一个环形中，如图12所示。每个芯片集群在一组不同的训练/评估输入上运行。在训练的情况下，在每个小批量迭代结束时，使用环来累积每个芯片集群生成的权重梯度，并分配更新后的权重。

轮圈结构最显著的优点是，它使FC层能够使用模型并行性(即FC层的参数跨芯片集群分割)进行并行化。FC层的特征和误差沿环形传递，并在适当的芯片簇上进行评估。模型并行性消除了沿环形发送FC层权值的需要，大大降低了环形带宽，也使流量更加均匀。此外，模型并行性增加了FcLayer芯片可以作为批处理的输入数量，因为与单个芯片集群相比，来自所有芯片集群的部分特征被收集到每个FcLayer芯片中。因此，模型的并行性降低了通过环传输的数据量，也降低了每个FcLayer芯片的带宽。

**4.编程**

开发一个编译器前端，以DNN拓扑和SCALEDEEP微架构配置作为输入，并使用SCALEDEEP ISA为设计中的每个CompHeavy tile生成代码。编译器分为两个阶段:(i)工作负载映射和(ii)代码生成。

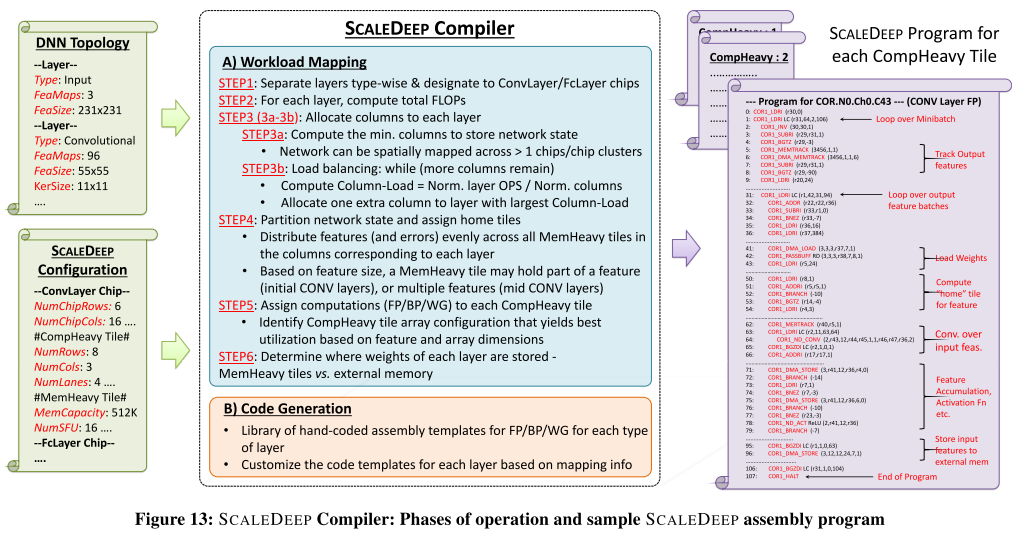
**负载映射**

工作负载映射阶段将芯片列分配到DNN中的每一层，并确定网络状态和计算如何分布到分配列的MemHeavy和CompHeavy块上。图13概述了工作负载映射阶段所涉及的步骤：

1. 首先，给定一个DNN拓扑结构，将CONV/SAMP层和FC层分离，并将它们分别指定在ConvLayer和FcLayer芯片上实现。
2. 计算了每个层所需的FLOP数量。
3. 为每一层分配芯片列。为此，首先计算每个层所需的最小列数，完全基于内存约束(STEP3a)。由于执行是流水线的，一个层的MemHeavy tiles应该累计包含特征和误差的两份副本，评估的部分特征/误差批处理的两份副本，以及相应的权重和权重梯度。基于最小列约束，确定了空间映射DNN所需的芯片/芯片集群的数量。然后，根据层的计算需求在芯片/芯片集群中分配额外的列。为此，计算每一层的每一列的负载，即归一化的触发器与分配给该层的归一化列数之比。附加列分配给列负载最高的层(STEP3b)。
4. 随着列的分配，网络状态均匀地分布在每个层对应的所有MemHeavy块上。在CONV层的情况下，根据特征大小，MemHeavy tiles可以包含一个或多个特征的一部分。
5. 然后，根据存储在连接的MemHeavy tiles中的特性，将计算分配给CompHeavy tiles的FP/BP/WG集。在此阶段，还将为CompHeavy tiles确定数组配置，以获得最佳的利用率。
6. 最后，根据每个层的可用内存容量，将权值和权梯度视为存储在芯片上或外部内存中。

**代码生成**

代码生成阶段为设计中的每个CompHeavy块生成SCALEDEEP程序。编译器使用一个手工编码的汇编例程模板库，用于每一层类型的FP/BP/WG步骤。编译器根据工作负载映射阶段提供的信息定制这些参数化的组装模板。



**5.实验方法**

