DOI: 10.3785/j.issn.1008-973X.2018.03.013

二维矩阵卷积的并行计算方法

张军阳,郭阳,扈啸
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要: 为了提高卷积神经网络模型中二维矩阵卷积的计算效率,基于 FT2000 多核向量处理器研究二维矩阵卷积的并行实现方法. 通过使用广播指令将卷积核元素广播至向量寄存器,使用向量 LOAD 指令加载卷积矩阵行元素,并通过混洗操作将不易并行化的矩阵卷积操作变成可以向量化的乘加操作,实现了通过减少访存、充分复用已取数据的方式来提高算法的执行效率. 设计卷积矩阵规模变化、卷积核规模不变和卷积矩阵规模不变、卷积核规模变化 2 种常用矩阵卷积计算方式,并对比分析不同计算方式对算法执行效率的影响. 基于服务器级多核 CPU 和 TI6678 进行实验对比,实验结果显示,FT2000 比多核 CPU 及 TI6678 具有更好的计算优势,相比多核 CPU 最高可加速 11 974 倍,相比 TI6678 可加速 21 倍.

关键词: 矩阵卷积;向量处理器;并行算法;性能优化;卷积神经网络
中图分类号: TP 391 **文献标志码**: A **文章编号**: 1008-973X(2018)03-0515-09

Parallel computing method for two-dimensional matrix convolution

ZHANG Jun-yang, GUO Yang, HU Xiao
(*College of Computer, National University of Defense Technology, Changsha* 410073, *China*)

Abstract: A parallel implementation method based on multi-core vector processor FT2000 was proposed to improve the computational efficiency of two-dimensional matrix convolution in convolution neural network model. The convolution kernel element was broadcast to vector register by using broadcast instruction; the row elements of the convolution matrix were vector loaded. With shuffle operation, the operation of matrix convolution, which is hard to be parallelled, can be vectorized by using multiply-add operation, and the implementation efficiency was achieved through reduction of access, full reuse of obtained data. Two kinds of common matrix convolution methods were designed: changing convolution matrix scale with constant convolution kernel size, and constant convolution matrix size with changing convolution kernel scale. The influence of different calculation methods on the algorithm execution efficiency was analyzed and compared. Finally, the comparison experiments were taken based on the server-level multi-core CPU and TI6678. Results show that FT2000 has a better computing advantage over multi-core CPU and TI6678, which can accelerate up to 11 974 times compared to multi-core CPU, while to TI6678 it is 21 times.

Key words: matrix convolution; vector processor; parallel algorithm; performance optimization; convolution neural network

收稿日期: 2017-03-04. **网址**: www.zjujournals.com/eng/fileup/HTML/201803013.htm
基金项目: 国家自然科学基金资助项目(60133007,61572025);国家重点研发计划资助项目(2016YFB0200401).
作者简介: 张军阳(1987—),男,博士生,从事体系结构、机器学习、嵌入式系统研究. orcid.org/0000-0002-2993-4494. E-mail:zhangjunyang11@nudt.edu.cn
通信联系人: 郭阳,男,教授. orci.org/0000-0003-1600-4666. E-mail: guoyang@nudt.edu.cn

以移动通信、互联网和智能终端等为代表的计算机和信息技术已在短短的几年间彻底改变了人类社会的生产和生活方式,人类已经进入到了大数据和智能科技时代.当下,从城市安防到社区监控,图像传感器已遍布城市的每个角落.数据时代不仅需要好的传感设备、海量存储和快速传播技术,还迫切需要对这些信息进行智能分类、识别和检索.让计算机能够像人类一样分析和理解真实世界中的图像一直是人类的梦想,然而现有的计算机图像理解技术无法满足这样的需求.

从2006年开始,深度学习(deep learning,DL)[1]技术开始受到学术界的广泛关注,在2012年举行的大规模图像识别挑战赛上,基于神经网络模型的AlexNet将传统图像识别错误率降低了41%,进而引发深度学习技术在工业界的研究热潮.当前基于深度学习的目标识别技术在图像识别[2]、语音识别[3]、自然语言处理等领域都取得了重大突破.深度学习由一系列神经网络模型组成,包括卷积神经网络(convolutional neural network,CNN)[4]、受限玻尔兹曼机(restricted Boltzmann machine,RBM)[5]、自动编码机[6]、循环神经网络(recurrent neural network,RNN)[7]等常用模型.其中基于卷积神经网络的深度学习模型是当前图像识别领域中的主力,在2012—2017年的大规模图像识别挑战赛(large scale visual recognition challenge,ILSVRC)中都取得了最佳识别率.高识别率同时带来了巨大的计算量,卷积神经网路模型中卷积层的计算量约占整个模型计算量的85%以上[8],因此,目前的许多研究大多是针对卷积神经网络的加速器,如GPU(graphic processing unit)[9-10]、FPGA(field programmable gated array)[11-12]、ASIC(application specific integrated circuit)[13-15]、向量DSP(digital signal processor)等.

向量处理器是一种具有多功能单元、多处理部件的新颖体系结构[16],其处理器内部包括面向流控处理和逻辑计算的标量处理部件和针对密集型大规模数据处理的向量处理部件,该向量处理部件内部集成了丰富的能够执行向量计算的向量处理单元,每个处理单元又具有若干个浮点乘加器和定点乘加器,支持单条指令完成浮点或定点的乘累加操作,能够提供强大的计算能力.然而针对新颖的微处理器结构,配套的软件开发工具也是一个重要的研究内容,如何通过软件编程的方式将多核向量处理器内部的所有硬件资源利用起来,发挥其多功能部件、多处理单元、充分利用各个层级的并行性,将不同的应用高效的映射是当前向量化程序面临的一大难题[17].

本文以二维矩阵卷积的计算为研究对象,面向多核向量处理器FT2000展开并行算法与性能优化研究.1)分析算法加速瓶颈并设计相应的并行方案;2)针对向量处理器的体系结构特点提出一种提高数据复用率的向量化实现方法;3)提出一种针对多核向量处理器的矩阵卷积并行实现方案;4)对比分析2种矩阵卷积计算模式对卷积实现性能的影响.

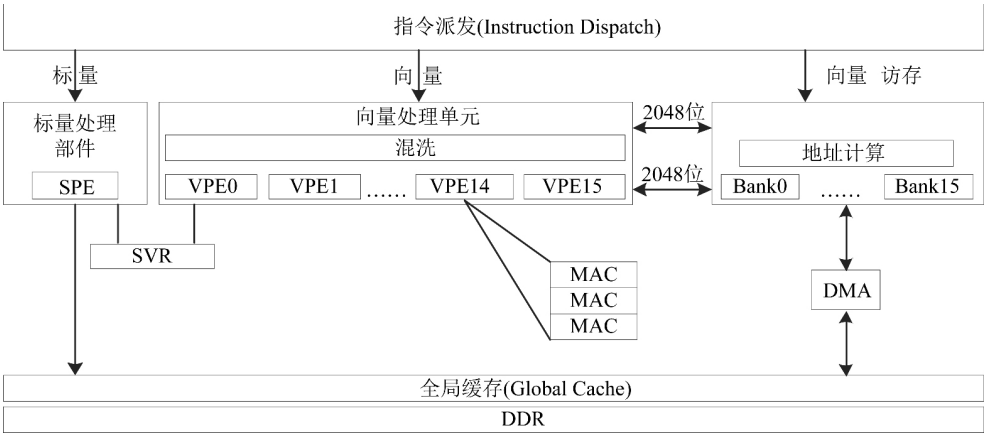


图1 FT2000处理器体系结构

Fig.1 Architecture of processor FT2000

1 向量处理器体系结构与优化策略

1.1 微处理器体系结构

FT2000是一种面向密集型浮点计算的12核向量处理器,单核结构如图1所示,12核的单精度浮点性能达2.4TFLOPS,12个核是一个同构的微处理器核通过环形互联网络连接起来,单核内部集成向量处理单元和标量处理器单元,标量单元负责控制和逻辑计算,向量单元主要面向密集型计算,其

包含 16 个向量处理单元，每个处理单元由若干浮点乘加单元和寄存器文件构成，所有的向量处理单元可以统一编址. 此外，向量处理单元和标量处理单元可以通过共享寄存器进行数据交互，并且通过广播操作支持标量寄存器到向量寄存器的广播操作. 向量处理单元可以同时发射 11 条指令，该执行包可以由若干条向量指令和若干条标量指令组成，指令派发部件可以对该执行包中的指令进行译码并指派到相应的功能单元去执行，标量处理单元具有本地标量存储体，主要存放需要标量来处理的数据，比如逻辑操作、地址计算等，因此，标量存储体空间较小，而向量处理单元主要存放需要进行大规模密集型计算的数据，因此，其存储体规模较大，标量存储体约 96 KB，向量存储体约 768 KB. 环形互联网络上面的 12 个处理器核可以通过共享 DDR 的方式通过 DMA 操作进行 DDR 与核内数据的相互传输，同时 DDR 也支持全局缓存，方便多核间的数据共享与交互.

1.2　优化策略

(1) 循环展开.

循环展开是一种常用的程序优化方法，可以通过减少循环体的循环次数，减少分支执行时间来为流水线提供更多的并行机会，是手工汇编优化程序的主要方法. 目前的编译器一般只对循环体很小的内循环进行展开，因此可能损失一些计算性能，尤其是对循环嵌套较多的循环体，编译器的优化效果不明显，因此可以通过手工汇编展开的方法来进行优化，由于 CNN 中的卷积计算具有多重循环体，对其核心代码采用手工汇编优化将大大提高程序执行效率.

(2) 指令级并行.

向量处理器有许多不同的功能单元，若能够利用其可以同时执行的特点，就可以大幅提高执行速度. 现代处理器将指令操作划分为不同的阶段，每个阶段由不同的单元执行，这样多个操作就可以在处理器的多个单元上流水执行，流水线技术也是向量处理器优化的一个重点. 此外，指令级并行还包括乱序执行、多发射、VLIW 和分支预测等.

(3) 向量化并行.

向量化[18]是指一条向量指令可以同时操作向量寄存器中的多个元素，是一种数据并行模式，也是向量处理器提升性能的重要方法，如 X86 的 SSE/AVX、ARM 的 NEON，都是基于 SIMD 模式. 另一种数据并行的方式是 SIMT，是 GPU 采用的向量化方法. 本文优化主要采用 SIMD.

(4) 缓存优化.

处理器的时钟频率和计算性能以超乎想象的速度增长，但是主存(DRAM)的访问速度的增长却缓慢的多，虽然 Cache 和预取能够减少平均访存时间，但仍不能从根本上解决问题，因此，在缓存优化过程中采用双 Buffer 机制，通过将计算时间和传输时间重叠，也可以大大提高程序的执行效率.

2　矩阵卷积算法介绍与分析

2.1　卷积神经网络的基本结构

如图 2 所示为一个典型 CNN 的基本结构，主要包括卷积层、池化层和全连接层，而其中卷积层的计算时间约占整个模型计算量的 85%，因此加速 CNN 模型中的卷积计算成为当前神经网络加速的一个研究热点.

2.2　二维矩阵卷积

二维卷积常用于图像处理中，给定一个图像 $X_{ij}(1\leqslant i\leqslant M, 1\leqslant j\leqslant N)$ 和滤波器 $f_{ij}(1\leqslant i\leqslant m, 1\leqslant j\leqslant n)$，一般 $m<M, n<N$，卷积的输出为

$$y_{ij}=\sum_{u=1}^{m}\sum_{v=1}^{n}(f_{uv}x_{i-u+1,\,j-v+1}). \tag{1}$$

2.3　二维矩阵卷积的实现方法分析

当前基于 GPU 的矩阵卷积实现主要有以下几种.

1) 将矩阵的卷积操作转化为矩阵乘法，然后通过使用已有的 GPU 矩阵乘法的优化算法进行计算，该方法需要将输入卷积矩阵 $\boldsymbol{D}$ 和卷积核 $\boldsymbol{F}$ 进行转化，由于该方法将 $\boldsymbol{D}$ 从四维数组转化为二维矩阵时数据规模变大，会占用较多的显存，且当卷积核越大，移动步长越小，显存的消耗就会越大.

2) 将输入特征图 $\boldsymbol{D}$ 和卷积核 $\boldsymbol{F}$ 首先进行 FFT 变换[19]，将数据变换到频域空间，然后将变换之后的结果在频域进行点乘运算，最后将点乘的结果进行 FFT 逆变换回时域空间，该方法的优点是计算量与卷积核的大小无关，因此针对卷积核尺寸较大的卷积网络，FFT 方法相比直接卷积方法在降低计算

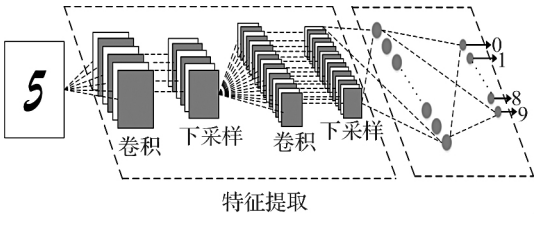


图 2　典型卷积神经网络结构

Fig. 2　Typical convolutional neural network (CNN) structure

量方面具有明显的优势,其不足之处就是要付出增加存储开销的代价,因此当前基于 FFT 的方法并没有得到广泛的应用,主要是因为当前的卷积神经网络模型中采用的卷积核尺寸都比较小,因此 FFT 方法在降低计算量方面的优势没能得到很好的展现.

3) 基于 cuDNN 的矩阵卷积算法实现[20-21],cuDNN 是 NVIDIA 公司针对 GPU 的众核架构高度优化的卷积神经网络加速库,主要针对矩阵卷积进行优化实现,其也是通过将卷积运算转化为矩阵乘法,不同的是转化后的二维输入特征图没有直接存储到显存里面,而是当需要这部分数据的时候,直接到 $\boldsymbol{D}$ 中索引对应的数据,并加载到 GPU 的显存里面进行计算,避免了占用额外显存的弊端.

传统的二维矩阵卷积计算过程如图 3(a)所示,文献[19]中用到了另一种计算方法,即将卷积矩阵和卷积核矩阵展开成矩阵块的方式进而通过普通矩阵相乘的方式进行卷积计算,如图 3(b)所示,其计算过程如下.

1) 根据卷积核规模和移动步长,将对应的卷积矩阵相应的卷积块展开成列向量的形式,所有展开的列向量组成一个新的矩阵.

2) 将单个卷积核矩阵按行展开成一个行向量,所有展开的卷积核组成一个新的卷积核矩阵.

3) 将新的卷积矩阵和新的卷积核矩阵做普通矩阵与矩阵乘法,进而可以通过调用普通的 BLAS (basic linear algebra subprograms)[23] 函数库中的 GEMM(general matrix multiplication)完成二维矩阵卷积的计算.

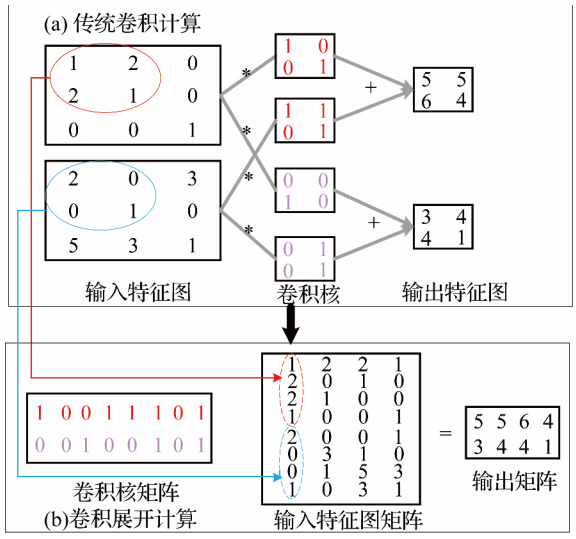


图 3 二维矩阵卷积计算的 2 种方式

Fig. 3 Two calculation methods for two-dimensional matrix convolution

可以看出,图 3(b)所示的计算方法的优点是将小的卷积计算模块组成大规模的计算单元,可以充分利用大规模矩阵乘法带来的高计算效率. 同样,该方法的缺点也非常明显,主要包括: 1) 在卷积计算前,需要将小规模卷积模块排列成大规模矩阵单元,而这些操作若不能并行执行,将带来不少计算开销. 2) 对卷积矩阵进行有重复的展开增加了矩阵的规模,随着卷积矩阵规模的增加,采用这种有重复展开的方式会造成数据规模越来越大,而对嵌入式处理器来说,核内存储最为重要,因此该方法不易于对核内存储敏感的向量处理器来执行.

此外,针对大规模二维矩阵卷积,文献[19]提出了一种分块计算的方法,即将大规模的矩阵根据卷积核的尺寸划分成合理的规模,然后对每个划分好的卷积模块采用与图 3(b)类似的方法进行展开,进而可以通过多个卷积模块的并行执行,完成大规模二维卷积矩阵的计算. 但该大规模二维矩阵卷积的分块方法,除了大大增加了卷积矩阵的数据存储量,还使得卷积计算的结果顺序上不连续,如图 4(b)所示,由于大部分二维卷积计算的结果需要作为后续操作的初始数据,比如 CNN,因此该方法也不利于后续数据的处理. 如图 4 所示为文献[19]提出的一种针对大规模二维矩阵卷积进行分块计算的方法.

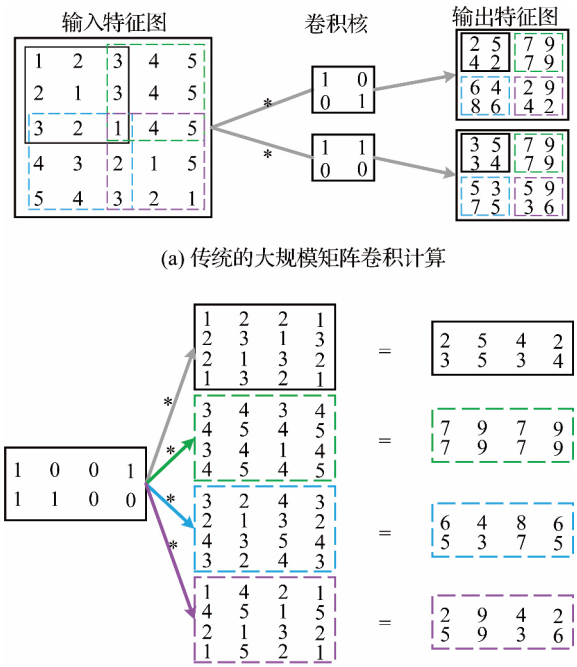


图 4 大规模矩阵卷积的子块并行方案

Fig. 4 Subblock parallel scheme for large scale matrix convolution

3　二维矩阵卷积的向量化设计与实现

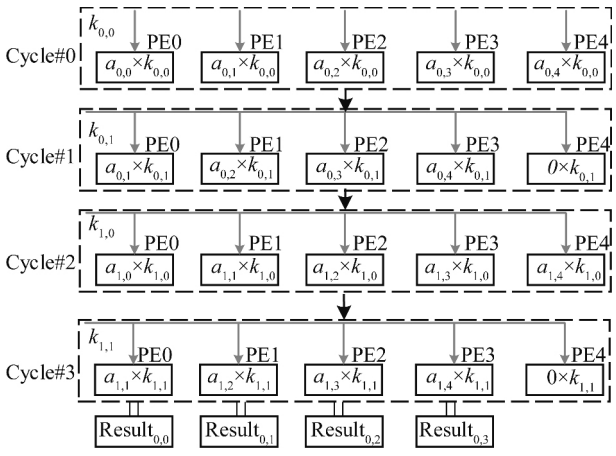
FT2000 是一款向量处理器，包含 768KB 的向量存储体 AM 和 96KB 的标量存储体 SM，AM 主要完成向量运算，SM 主要完成相应的标量操作，一般来说卷积核矩阵的规模都比较小，当下主流 CNN 中的卷积核规模一般为 3×3、5×5、7×7、11×11. 且根据算法的需要，卷积核数据需要标量取，因此将卷积核数据置于 SM，卷积矩阵置于 AM，其向量化实现过程如下.

3.1　单核程序设计分析

(1) 当卷积矩阵规模较小时，即卷积矩阵可以全部置于 AM 中，以单精度浮点数据为例，单个浮点数据为 4 byte，设卷积矩阵为 $N\times N$ 的方阵，则 $N\times N\times 4=768\times 1\ 024$，AM 中可以放置的最大卷积矩阵的规模为 443×443，因此当需要计算的卷积矩阵规模大于 443×443 时，需要从片外加载数据.

下面以图 5 为例说明基于向量处理器的矩阵卷积的实现方法. 设卷积矩阵 $\boldsymbol{A}$ 为 5×5，卷积核矩阵为 2×2，滑动步长为 1，边界扩充操作为 0，其卷积结果第一行元素的计算过程如 Cycle ＃0 至 Cycle ＃3 所示.

Cycle ＃0：所有 PEs 加载卷积矩阵 $\boldsymbol{A}$ 的第一行元素($a_{0,0}$，$a_{0,1}$，$a_{0,2}$，$a_{0,3}$ 和 $a_{0,4}$)，标量处理单元加载卷积核矩阵的第 1 行第 1 个元素($k_{0,0}$)，通过标向量广播将 $k_{0,0}$ 广播至向量寄存器中，即对应的向量寄存器中的元素为 $k_{0,0}$，$k_{0,0}$，$k_{0,0}$，$k_{0,0}$，$k_{0,0}$，通过向量处理器的乘加指令，每一个 PE 完成对应元素与卷积核 $k_{0,0}$ 的乘法，并将乘法结果累加至对应的向量寄存器中.



(A：5×5；步长：1×1；kernel：2×2；pad：0)

图 5　FT2000 矩阵卷积的算法映射

Fig. 5　Algorithm mapping of FT2000 matrix convolution

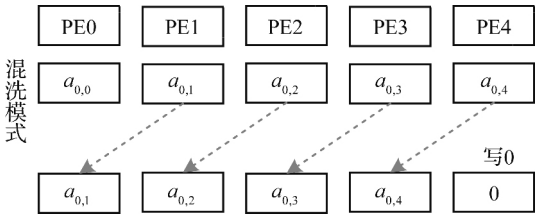


图 6　FT2000 移位的混洗模式

Fig. 6　Shift shuffle mode of FT2000

Cycle ＃1：通过配置特殊的混洗模式(如图 6 所示)将 Cycle ＃0 中加载的行元素进行移位，完成移位后当前 PEs 所对应的元素为 $a_{0,1}$，$a_{0,2}$，$a_{0,3}$，$a_{0,4}$，0，同时标量加载卷积核的第 1 行第 2 个素 $k_{0,1}$，同样通过标向量广播至向量寄存器中，即为 $k_{0,1}$，$k_{0,1}$，$k_{0,1}$，$k_{0,1}$，$k_{0,1}$，通过向量乘加指令，每一个 PE 完成对应元素与卷积核 $k_{0,1}$ 的乘法，并将乘法结果累加至 Cycle ＃0 中的累加寄存器中.

Cycle ＃2：所有 PEs 加载卷积矩阵 $\boldsymbol{A}$ 的第二行元素($a_{1,0}$，$a_{1,1}$，$a_{1,2}$，$a_{1,3}$ 和 $a_{1,4}$)，标量处理单元加载卷积核矩阵的第 2 行第 1 个元素($k_{1,0}$)，通过标向量广播将 $k_{1,0}$ 广播至向量寄存器中，即对应向量寄存器中的元素为 $k_{1,0}$，$k_{1,0}$，$k_{1,0}$，$k_{1,0}$，$k_{1,0}$，通过向量处理器的乘加指令，每一个 PE 完成对应元素与卷积核 $k_{1,0}$ 的乘法操作，并将乘法结果累加至 Cycle ＃1 中的累加寄存器中.

Cycle ＃3：采用 Cycle ＃1 中的混洗模式，将 Cycle ＃2 中取到的元素进行移位，完成移位后当前 PEs 所对应的元素为 $a_{1,1}$，$a_{1,2}$，$a_{1,3}$，$a_{1,4}$，0，同时标量加载卷积核矩阵第 2 行第 2 个元素并广播至向量寄存器中，即为 $k_{1,1}$，$k_{1,1}$，$k_{1,1}$，$k_{1,1}$，$k_{1,1}$，通过向量处理器的乘加指令，每一个 PE 完成对应元素与对应卷积核 $k_{1,1}$ 的乘法操作，并将乘法结果累加至 Cycle ＃2 中的累加寄存器中. Cycle ＃3 累加寄存器中的值即为卷积结果矩阵第一行的结果元素.

通过以上计算过程可以发现，采用本文提出的方法有以下几个优点：1) 在进行卷积计算前不需要额外的开销将卷积矩阵进行展开；2) 大幅减少了矩阵展开所带来的存储开销；3) 可以充分复用已取到的数据，减少数据的访存量，进而节约计算时间，因为取数据比数据的计算要花费更多的节拍；4) 通过流水线技术，理想情况下每拍可以同时计算结果矩阵的一行元素；5) 不仅不需要将卷积矩阵进行展开，也避免了中科院计算所陈云霁等[15]提出的 PEs 阵列间数据的水平方向或垂直方向频繁的数据移动.

(2) 当卷积矩阵规模较大时,即 $N\times N\times 4>768\times 1024$ 时,卷积矩阵 $\boldsymbol{A}$ 的规模大于 443×443 时,由于 AM 一次不能加载整个卷积矩阵,为了提高卷积计算的速度,采用双 Buffer 机制,用"ping-pong"的方式进行数据传输,此时 AM 空间按地址划分为相等的两部分,分别为 Buffer0 和 Buffer1,数据从 DDR 中通过 DMA 传输到核内 AM 中,假设从 Buffer0 开始计算,Buffer0 计算的同时,Buffer1 进行数据传输,当 Buffer1 开始计算时,Buffer0 输出上一次的计算结果并启动新数据传输,采用双 Buffer 的机制可以将计算时间和传输时间重叠起来,提高算法的执行效率. 如图 7 所示为本文采用的双 Buffer 机制示意图. 当处理大规模矩阵卷积时在采用双 Buffer 机制的同时,内核还是采用(1)中的向量化实现方法来处理.

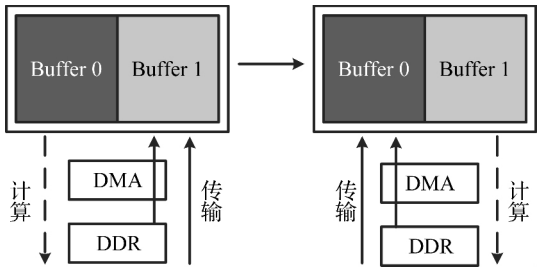


图 7 FT2000 双 Buffer 机制示意

Fig. 7 Double Buffer mechanism of FT2000

3.2 多核程序设计分析

由于 FT2000 是一款 12 核高性能向量处理器,采用多核并行的技术实现高效的算法加速也是一个重要的研究内容. 但是多核程序的设计与单核程序不同,需要考虑多方面的因素,比如如何进行多核的任务划分、算法是否有很强的相关性、多核的通信开销如何等等. 考虑到二维矩阵卷积主要用于图像处理尤其是卷积神经网络中卷积层的计算,且卷积层的计算规模从 5×5 到 $1\ 000\times 1\ 000$ 都存在,大部分都是卷积核规模较小的矩阵卷积计算,因此针对矩阵卷积的多核并行一般不会将单个卷积的计算分配给多核来完成,而是 N 个独立的卷积计算过程由 N 个核并行执行来完成卷积计算的多核并行.

如图 8 所示为 FT2000 的多核实现方案,从 DDR 中获得一幅输入特征图像,并广播至所有 N_c 个核共有,同时每个核单独加载自己的卷积核矩阵至标量存储体 SM 中,则多核程序中每个核的计算过程同 3.1 单核程序的计算一样.

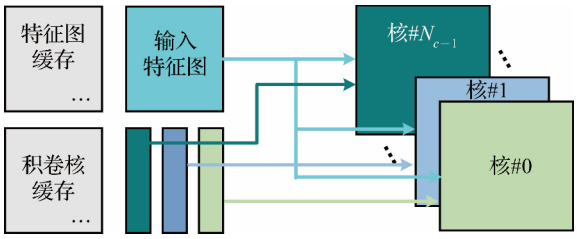


图 8 矩阵卷积多核实现方案

ig. 8 Multi-core implementation of matrix convolution

4 实验结果与分析

4.1 实验平台

本实验的对比平台有服务器级 CPU-Intel(R) Xeon(R) CPU E5-2650,32 核,64 GB 内存,主频为 2.6 GHZ,软件平台使用 Linux CentOS6.2 操作系统、使用 C 语言实现不同规模的矩阵卷积,基于 GCC4.8.2 编译器并使用最高级别优化(-O3)优化所有代码程序,测试时间由 clock() 函数进行统计;使用 TI 的高性能多核数字信号处理器 TMS320C6678,8 核,1.25 GHZ 主频,每个核拥有 32 KB 的 L1P 和 32 KB 的 L1D,并基于 CCS5.5 软件编程平台完成所有程序代码的测试与性能统计;飞腾平台使用类 CCS 的 FT2000 软件编程平台并完成 FT2000 向量程序代码的编写与性能统计.

4.2 算法加速比

通过 2.2 节的分析可知,二维矩阵卷积是典型的计算密集型和访存密集型算法,而当前的 CNN 网络模型中矩阵卷积的计算又占据着整个模型计算量的 85%以上,因此本实验选取 2 种计算模型进行分析. 第一种,卷积矩阵不变,卷积核矩阵规模相应增加,记为模式 1;第二种,卷积核矩阵不变,卷积矩阵规模相应增加,记为模式 2.(本实验中所有数据采用双精度浮点,且都为方阵,即 48×3 表示 48×48 与 3×3 的卷积操作).

如图 9 所示为基于卷积核规模变化的 FT2000 优化前与优化后的加速比折线图,S_P 为加速比. 从图中可以看出,当卷积矩阵规模一定时,矩阵卷积计算的优化加速比随着卷积核规模的增加基本呈线性增加趋势. 如图 10 所示为基于卷积矩阵规模变化的 FT2000 优化前与优化后的加速比折线图. 可以看出,当卷积核规模一定时,算法加速比随着卷积矩阵规模的增加先降低,后趋于稳定.

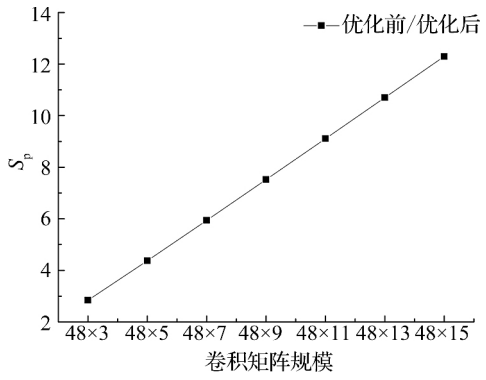


图 9　基于卷积核规模变化的 FT2000 优化加速比

Fig. 9　Optimized speed-up ratio of FT2000 with change of convolution kernel scale

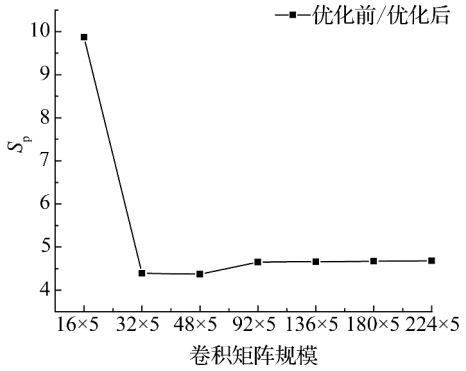


图 10　基于卷积矩阵变化的 FT2000 优化加速比

Fig. 10　Optimized speed-up ratio of FT2000 with change of matrix scale

图 11 统计了随着卷积核规模的变化，FT2000 基于 CPU 的加速比，可以看出 FT2000 向量处理器相对于服务器级多核 CPU 取得了 2132～11974 倍不等的加速比. 图 12 为卷积核规模不变，随着卷积矩阵规模变化的 FT2000 基于 CPU 的加速比，同样取得了 456～7925 倍不等的加速比. 这说明对于计算密集型的二维矩阵卷积 FT2000 向量处理器比多核 CPU 并行系统能够获得更好的性能.

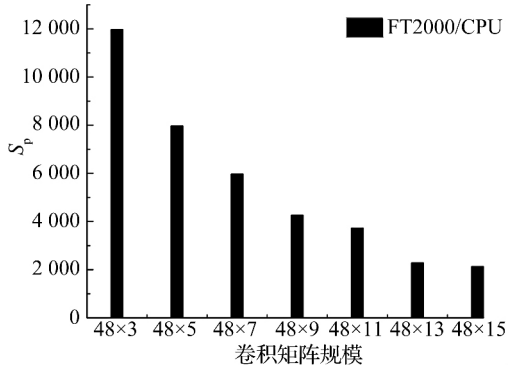


图 11　基于卷积核规模变化的 FT2000/CPU 的加速比

Fig. 11　Speed-up ratio of FT2000/CPU with change of convolution kernel scale

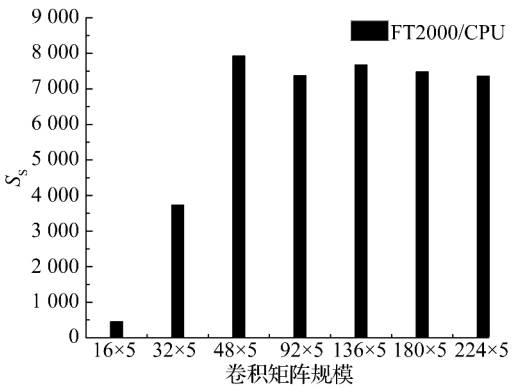


图 12　基于卷积矩阵变化的 FT2000/CPU 的加速比

Fig. 12　Speed-up ratio of FT2000/CPU with change of matrix scale

图 13、图 14 针对 2 种计算模式，分别对比分析了 FT2000 平台基于 TI6678 取得的性能加速比情况. 在图 13 中，FT2000 基于 TI6678 的加速比随着模式 1 计算规模的增加所取得的优势愈发明显，而图 14 中，当模式 2 的计算规模增加到一定程度，其相对 TI6678 的加速比基本维持在 20 倍左右. 可见，此 2 种计算模式，对程序的性能优化影响十分明显.

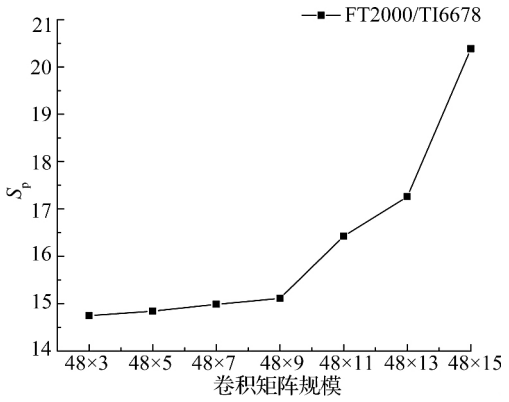


图 13　基于卷积核规模变化的 FT2000/TI6678 的加速比

Fig. 13　Optimized speed-up ratio of FT2000/ TI6678 with change of convolutional kernel scale

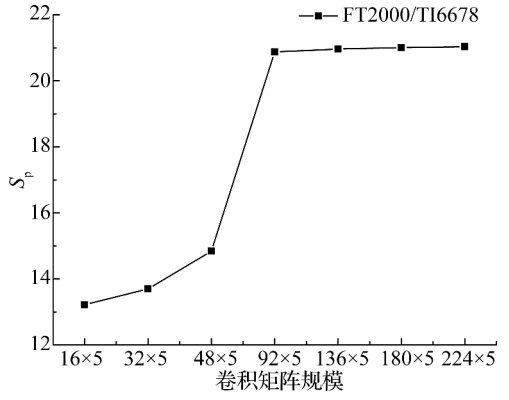


图 14　基于卷积矩阵变化的 FT2000/TI6678 的加速比

Fig. 14　Optimized speed-up ratio of FT2000/ TI6678 with change of matrix scale

4.3 算法性能瓶颈分析

图9、图10显示了2种不同的计算模式对算法优化加速比的影响,当卷积核规模增加时,加速比基本成线性增加,而当卷积矩阵规模增加时,加速比有所下降,并最终保持稳定,且加速比并不高,这主要是因为在该算法实现过程中,内核程序主要是通过卷积核的规模来控制循环大小,即当卷积核规模较小时,程序的最内层循环较小,无法通过软件流水的方式高度优化矩阵卷积计算,进而难以发挥向量处理器的计算优势,而图9中之所以取得线性加速比,主要是因为卷积核的规模增加,进而核心循环可以通过软件流水等方式对实现代码进行高度优化,因此,卷积核的规模是影响本文算法实现性能的一个主要影响因素.

图11中,随着卷积核计算规模的增加FT2000并没有取得线性的加速比,且有所下降,可能原因是随着矩阵计算规模的增加,CPU的多核及多线程发挥了作用,提高了卷积的计算效率.图12中的加速比相对稳定,主要原因是卷积核规模较小,只有卷积矩阵规模的增加并不能充分发挥FT2000向量处理器的作用.而图13中的计算模式1之所以能够取得较好的加速比,主要是因为FT2000算法优化中卷积核规模决定着内层循环的大小,内层循环越大,通过循环展开所取得的性能优势就愈发明显.图14中之所以没有取得较好的加速比,其瓶颈仍在于卷积核规模过小,使得FT2000向量处理器没有完全发挥其性能优势.针对该计算瓶颈,下一步可考虑能否在不增加卷积矩阵规模的前提下,通过融合多个卷积核矩阵来提高FT2000向量处理器计算矩阵卷积的计算效率.

5 结 语

本文在对矩阵卷积计算深入分析的基础上,结合FT2000多核向量处理器的体系结构特点提出了一种减少访存、充分复用已取数据的二维矩阵卷积并行计算方法,并基于多核CPU和TI6678进行了性能对比和分析,取得了良好的加速比,实验结果显示:FT2000比CPU及TI6678具有更好的计算优势,相比CPU最高可加速11 974倍,相比TI6678可加速21倍.本文研究显示,二维矩阵卷积可在FT2000向量处理器上取得良好的性能,切实加速了矩阵卷积的计算过程,为今后国产FT2000高性能多核向量处理器用于卷积神经网络模型的移植作准备.

参考文献(References):

[1] DENG L, YU D. Deep learning: methods and applications [J]. **Foundations & Trends© in Signal Processing**, 2014, 7(3): 197-387.

[2] WU M, CHEN L. Image recognition based on deep learning[C] // **Chinese Automation Congress**. Wuhan: IEEE, 2015.

[3] LI D, LI J Y, HUANG J T, et al. Recent advances in deep learning for speech research at Microsoft [C]//**In the Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing**. Vancouver: IEEE, 2013: 8604-8608.

[4] KAVUKCUOGLU K, BOUREAU Y L, BOUREAU Y L, et al. Learning convolutional feature hierarchies for visual recognition [C] // **International Conference on Neural Information Processing Systems**. Vancouver: Curran Associates Inc. 2010: 1090-1098.

[5] CHEN Z, WANG J, HE H, et al. A fast deep learning system using GPU [C] // **Proceedings of International Symposium on Circuits and Systems**. Melbourne: IEEE, 2014: 1552-1555.

[6] BOURLARD H, KAMP Y. Auto-association by multilayer-perceptrons and singular value decomposition [J]. **Biological Cybernetics**, 1988, 59(4/5): 291-294.

[7] YAJIE MIAO, MOHAMMAD GOWAYYED, AND FLORI-AN METZE. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding [C] // **Automatic Speech Recognition and Understanding**. Scottsdale: IEEE, 2015: 167-174.

[8] LIU S, DU Z, TAO J, et al. Cambricon: an instruction-set architecture for neural networks [J]. **ACM Sigarch Co-mputer Architecture News**, 2016, 44(3): 393-405.

[9] NASSE F, THURAU C, FINK G A. Face detection using GPU-based convolutional neural networks [C] // **International Conference on Computer Analysis of Images and Patterns**. Berlin Heidelberg: Springer, 2009: 83-90.

[10] POTLURI S, FASIH A, VUTUKURU L K, et al. CNN-based high performance computing for real time image-processing on GPU [C] // **The Workshop on Nonlinear Dynamics & Synchronization & Int'l Symposium on Theoretical Electrical Engineering**. Klagenfurt: IEEE, 2011: 1-7.

[11] YU Q, WANG C, MA X, et al. A deep learning prediction process accelerator based FPGA [J]. **Proceedings of the Annual ACM Symposium on Theory of Computing**, 2015: 585-594.

[12] HEGDE G, SIDDHARTHA, RAMASAMY N, et al. Evaluating embedded FPGA accelerators for deep

learning applications［C］// **IEEE，International Symposium on Field Programmable Custom Computing Machines**. Washington DC.：IEEE，2016：25.

[13] CHEN T，DU Z，SUN N，et al. DianNao：a small-footprint high throughput accelerator for ubiquitous machine learning［J］. **ACM Sigarch Computer Architecture News**，2014，49(4)：269-284.

[14] LIU D，CHEN T，LIU S，et al. PuDianNao：a polyvalent machine learning accelerator［C］// **Twentieth International Conference on Architectural Support for Programming Languages and Operating Systems**. Istanbul：ACM，2015：369-381.

[15] DU Z. ShiDianNao：shifting vision processing closer to the sensor［C］// **ISCA'15 Proceedings of the，International Symposium on Computer Architecture**. Portland：ISCA，2015：92-104.

[16] **刘仲，田希，陈磊. 支持原位计算的高效三角矩阵乘法向量化方法**［J］. **国防科技大学学报**，2014，6(36)：7-11.
LIU Zhong，TIAN Xi，CHEN Lei. Efficient vectorization method of triangular matrix multiplication supporting in-place calculation［J］. **Journal of National University of Defense Technology**，2014，6(36)：7-11.

[17] **刘仲，陈跃跃，陈海燕. 支持任意系数长度和数据类型的** FIR **滤波器向量化方法**［J］. **电子学报**，2013，2(41)：346-351.
LIU Zhong，CHEN Yue-yue，CHEN Hai-yan. A vectorization of fir filter supporting arbitrary coefficients length and data types［J］. **Acta Electronica Sinica**，2013，2(41)：346-351.

[18] **周海芳，高畅，方民权. 基于** CUBLAS **和** CUDA **的** MNF **并行算法设计与优化**［J］. **湖南大学学报：自科版**，2017，4(44)：147-156.
ZHOU Hai-fang，GAO Chang，FANG Min-quan，Parallel algorithm design and performance optimization of maximum noise fraction rotation based on CUBLAS and CUDA［J］. **Journal of Hunan University：Natural Sciences**，2017，4(44)：147-156.

[19] LAVIN A，GRAY S. Fast algorithms for convolutional neural networks［J］. **Computer Science**，2015：4013-4021.

[20] ZAGORUYKO S，KOMODAKIS N. Learning to compare image patches via convolutional neural networks［C］// **Computer Vision and Pattern Recognition**. Boston：IEEE，2015：4353-4361.

[21] POTLURI S，FASIH A，VUTUKURU L K，et al. CNN based high performance computing for real time image processing on GPU［C］// **Nonlinear Dynamics and Synchronization**. Klagenfurt.：IEEE，2011：1-7.

[22] CHELLAPILLA K，PURI S，SIMARD P. High performance convolutional neural networks for document processing［C］// **Tenth International Workshop on Frontiers in Handwriting Recognition**，La Baule：Suvisoft，2006：inria-00112631.

[23] DONGARRA J J. An extended set of FORTRAN basic linear algebra subprograms［J］. **ACM Transactions on Mathematical Software**，1988，14(1)：18-32.