深度神经网络DNN程序在CPU+MIC异构集群上的优化

邱君仪 @ ITOC

2016.5.14

大纲

- 背景介绍
- 单节点优化
- 集群并行优化
- 总结

背景

- 在语音识别领域中，深度学习已成为一种强有力的技术
- 深度神经网络在语音识别领域的应用大幅度地提高了已有语音识别系统的准确率，使得语音识别技术变得更加可用
- 深度学习中存在大量的矩阵计算、卷积计算，能通过异构并行加速获得极高的性能提升

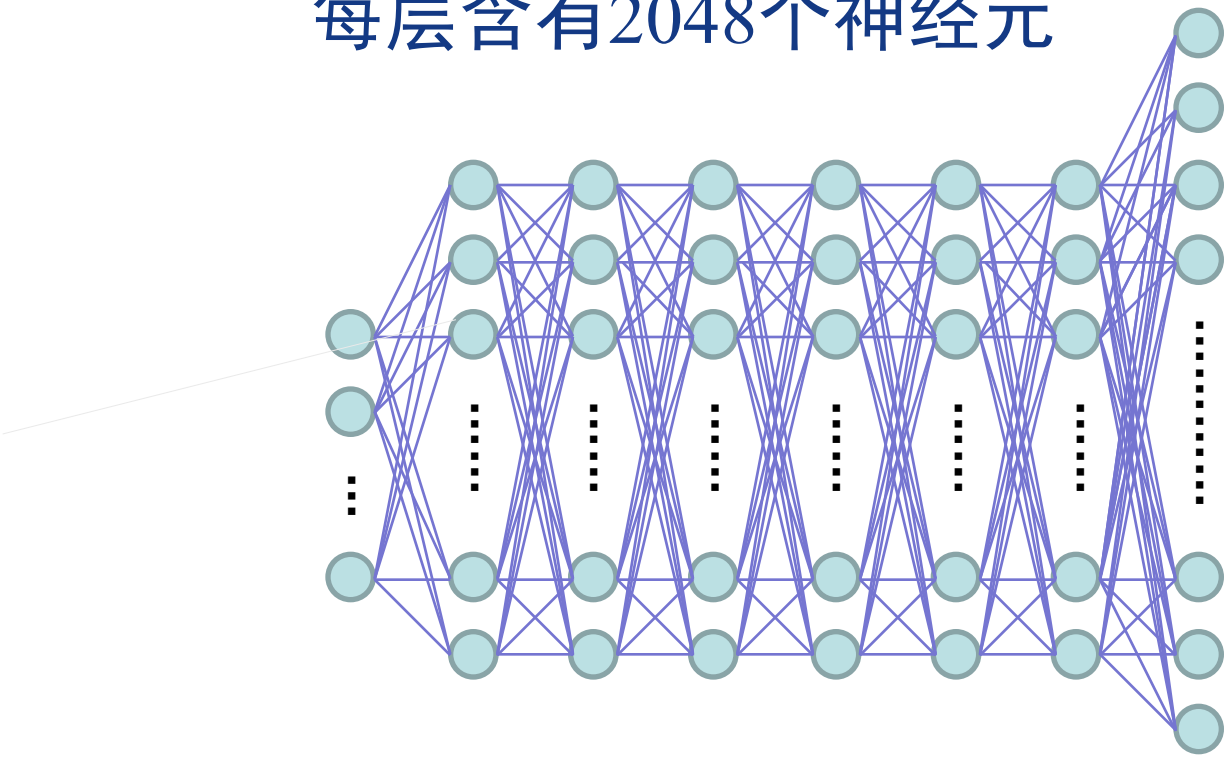
项目概要

- 针对语音识别中的声学分析模型而建立的深度神经网络训练程序

- 不同于常见的Caffe等通用架构，本应用是由科大讯飞开发的一个精简的、用于特定情形的DNN训练程序，以实现简易性、程序效率优先。

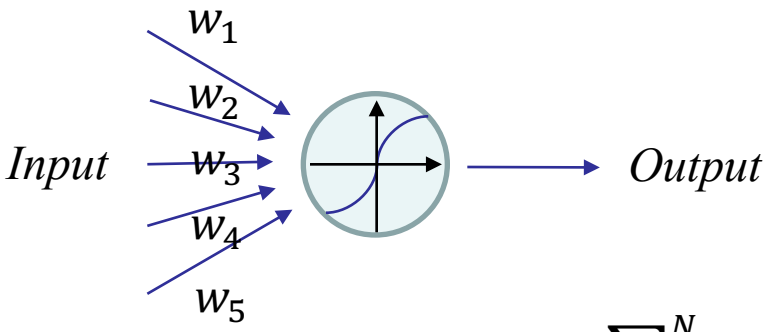
网络结构

- 输入层：429个特征
- 输出层：约9000个特征类
- 隐藏层：6个全连接(full connected)层，每层含有2048个神经元



网络结构

- 输入层：429个特征
- 输出层：约9000个特征类
- 隐藏层：6个全连接(full connected)层，每层含有2048个神经元
- 激活函数：sigmoid函数



$$Output = Sigmoid\left(bias + \sum_0^N input_i \cdot w_i\right)$$

网络结构

- 输入层：429个特征
- 输出层：约9000个特征类
- 隐藏层：6个全连接(full connected)层，
  每层含有2048个神经元
- 激活函数：sigmoid函数
- 训练模式：前向后向(forward-backward)算法
- 收敛算法：基于mini-batch的SGD方法

程序结构

- 程序核心通过过反向传播算法训练模型，分为：
  - Forward
    - 根据网络连接权重计算输入：matrix multiplication
    - 计算神经元输出：Sigmoid activation (kernel func.)
    - 输出层：Softmax重整化 (kernel func.)
  - Backward
    - 神经元反传误差：error propagation (kernel func.)
    - 连接反传误差：matrix mul.
  - Update
    - 计算梯度并更新模型： matrix mul. & (kernel func.)

大纲

- 背景介绍
- 单节点优化
- 集群并行优化
- 总结

单节点优化

- 编译优化
  - 使用Intel C/C++ Compiler
  - 调整编译选项，来更好地利用硬件性能
- 使用Intel MKL库
  - 程序热点为大量的矩阵运算(cblas_sgemm)
  - 多线程版的MKL库函数能够显著提升计算性能

- 这些优化手段带来近3倍的性能提升

单节点优化

- 线程级并行、数据级并行
  - 存在多个kernel函数对矩阵进行逐项操作，如：
    - 正向过程中的激活函数：

```
for i in range(0, Batch_Size):
    for j in range(0, Neurons_number):
        Output[i][j] = 1 / ( 1 + exp ( -Input[i][j] ) )
```

    - 反向过程中的误差传播：

```
for i in range(0, Batch_Size):
    for j in range(0, Neurons_number):
        InErr[i][j] = OutErr[i][j] * Output[i][j] * ( 1 - Output[i][j] )
```

  - 外层循环按样本遍历，用OpenMP做多线程并行
  - 内层循环按神经元遍历，通过向量化指令做数据并行
  - 进一步，性能提升约40%

单节点优化

- 内存访问优化
  - 去除大量冗余内存的复制
  - 使用cache blocking提升矩阵数据的利用率
    - 如，在批训练时将多个样本的误差归约到参数梯度

```
for j in range(0, Neurons_number):
    for i in range(0, Batch_Size):
        Gradient[j] = Gradient[j] + Error[i][j]
```

    - 通过循环换序、循环拆分，充分利用缓存数据：

```
for j1 in range(0, Neurons_number/Width):
    for i in range(0, Batch_Size):
        for j2 in range(0, W=Width):
            Gradient[j1*W+j2] = Gradient[j1*W+j2] + Error[i] [j1*W+j2]
    for j2 in range(0, Width):
        Parameter[j1*W+j2] = Parameter[j1*W+j2] – Gradient[j1*W+j2] * learning_rate
```

  - 进一步，加速6-7%

单节点优化

- 使用Xeon Phi卡加速
  - 在单节点中使用一块Xeon Phi协处理器
  - 将热点部分分载到Xeon Phi卡上（offload模式）
  - 通过数据的驻留，减少CPU与Phi间的数据通信
  - MKL、访存优化等手段能够很好的适用于Xeon Phi
  - 进一步，加速约25-35%

单节点优化

- 整体上，单节点加速接近7倍

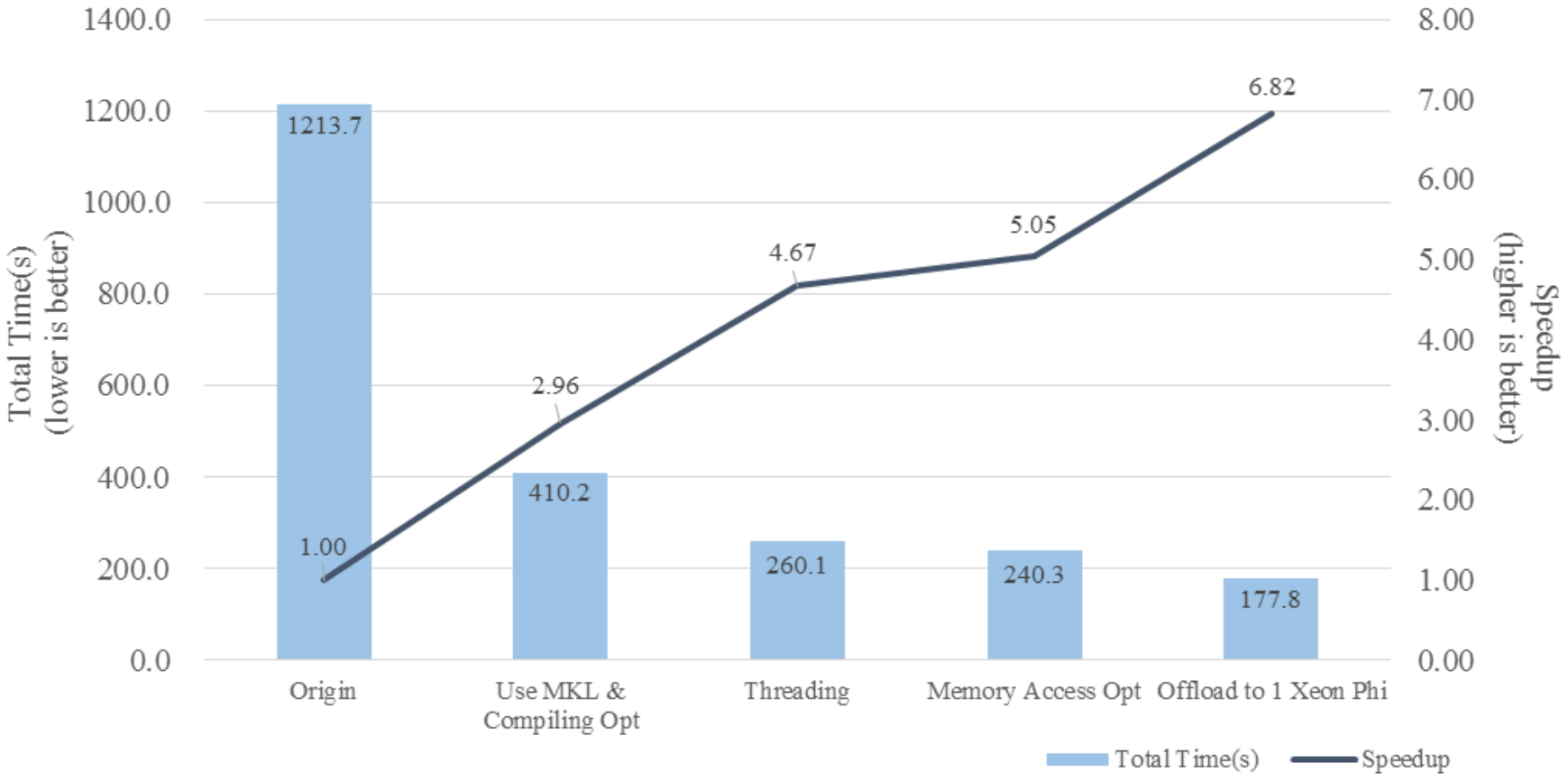


Fig. Intra-Node Performance Improvement Comparison On Testing Case

大纲

- 背景介绍
- 单节点优化
- 集群并行优化
- 总结

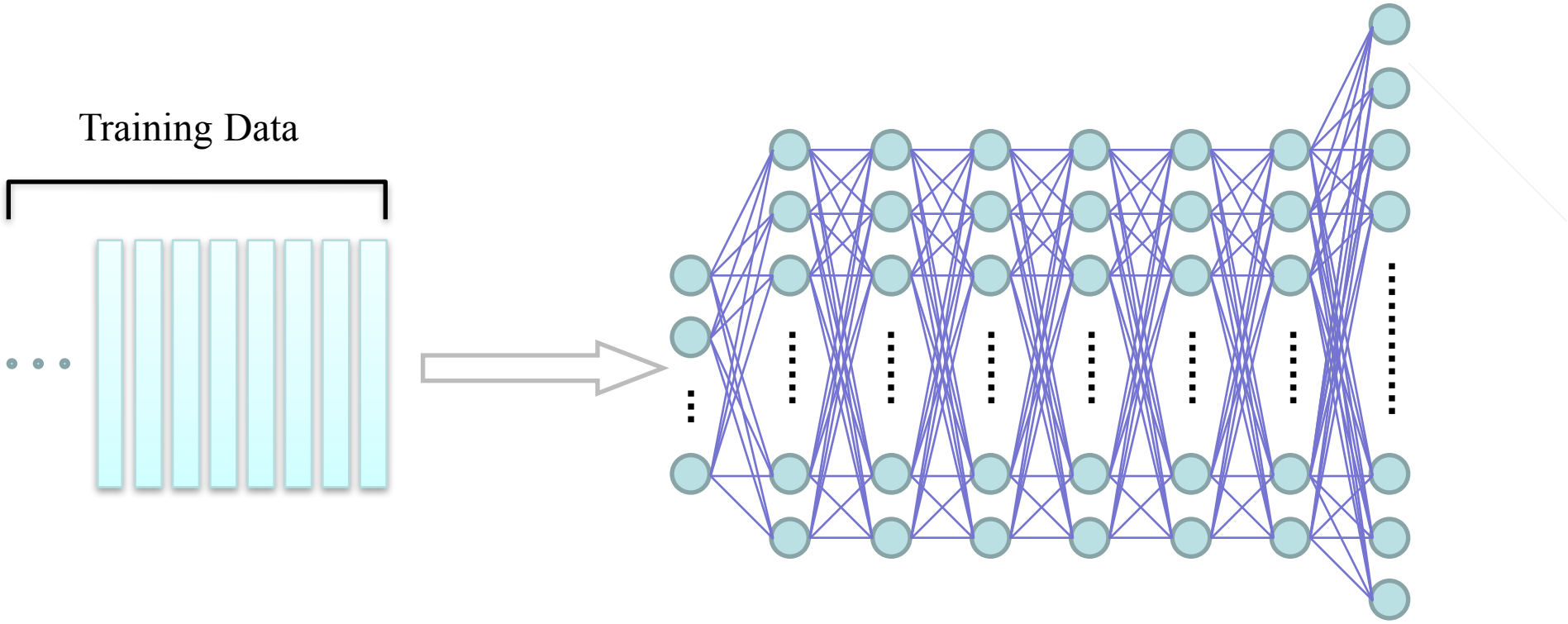
集群并行优化

- 目标：在不影响网络模型的精度下，将DNN训练程序扩展到天河2号集群的8个节点上（每个节点有3块Xeon Phi）
- 原因：单一训练进程需要相当长的时间用大量样本数据来训练网络模型
- 难点：
  - 训练网络的扩展性与网络模型的精度两者相互矛盾
    这在全连接网络中更加突出
  - 常见的神经网络并行训练方案会导致精度下降

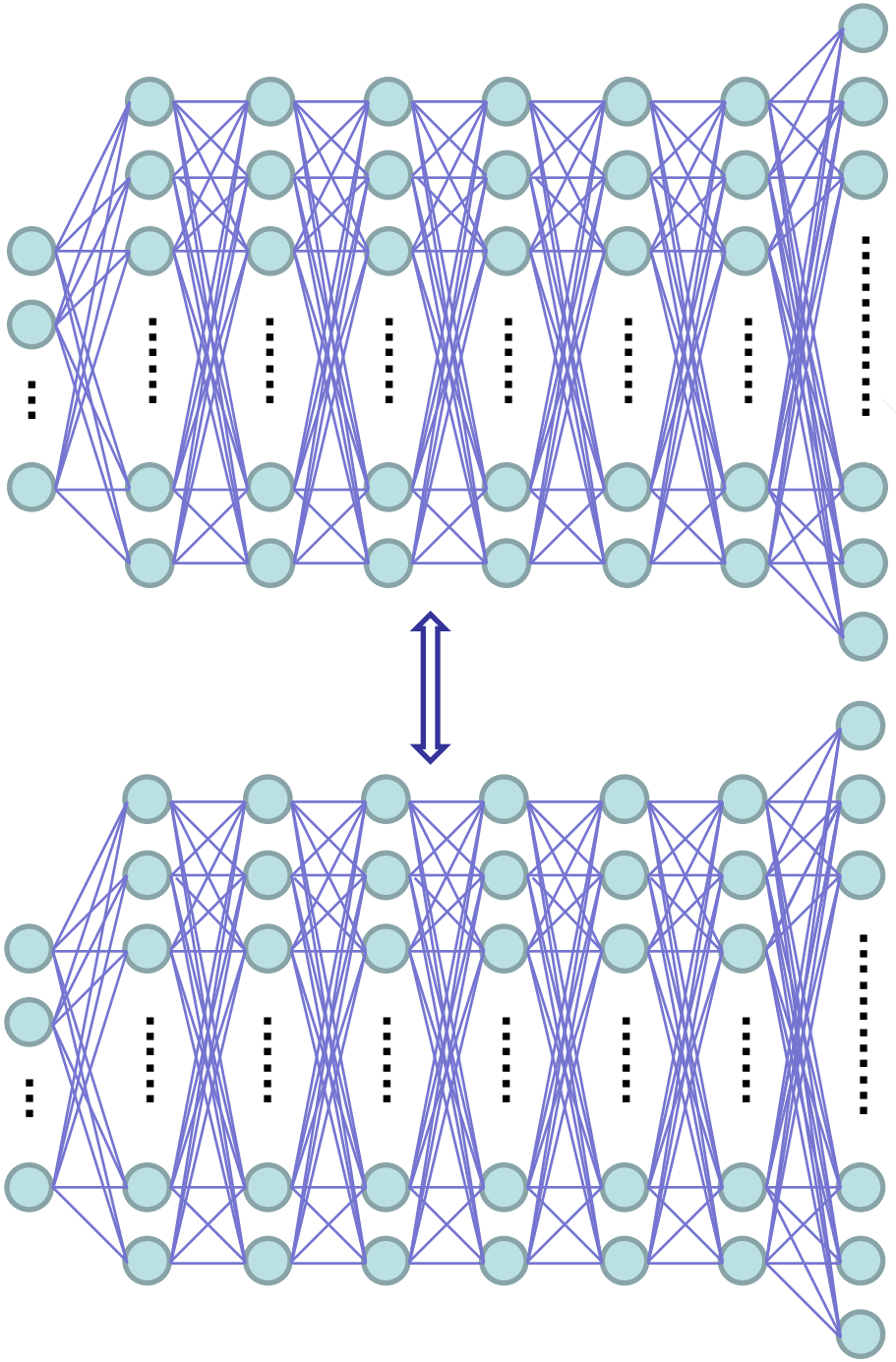
常见并行训练方法

- 数据并行
  - 优点：扩展性较好，容易实现
  - 缺点：频繁的模型同步会带来大量通信
    减小同步频率，模型精度下降

单一训练进程



多训练进程并行



常见并行训练方法

- 数据并行
  - 优点：扩展性较好，容易实现
  - 缺点：频繁的模型同步会加重通信
    减小同步频率，模型精度下降
- 模型并行
  - 仅适用于局部连接(locally connected)的网络
  - 对全连接网络将产生大量的进程通信

集群并行优化

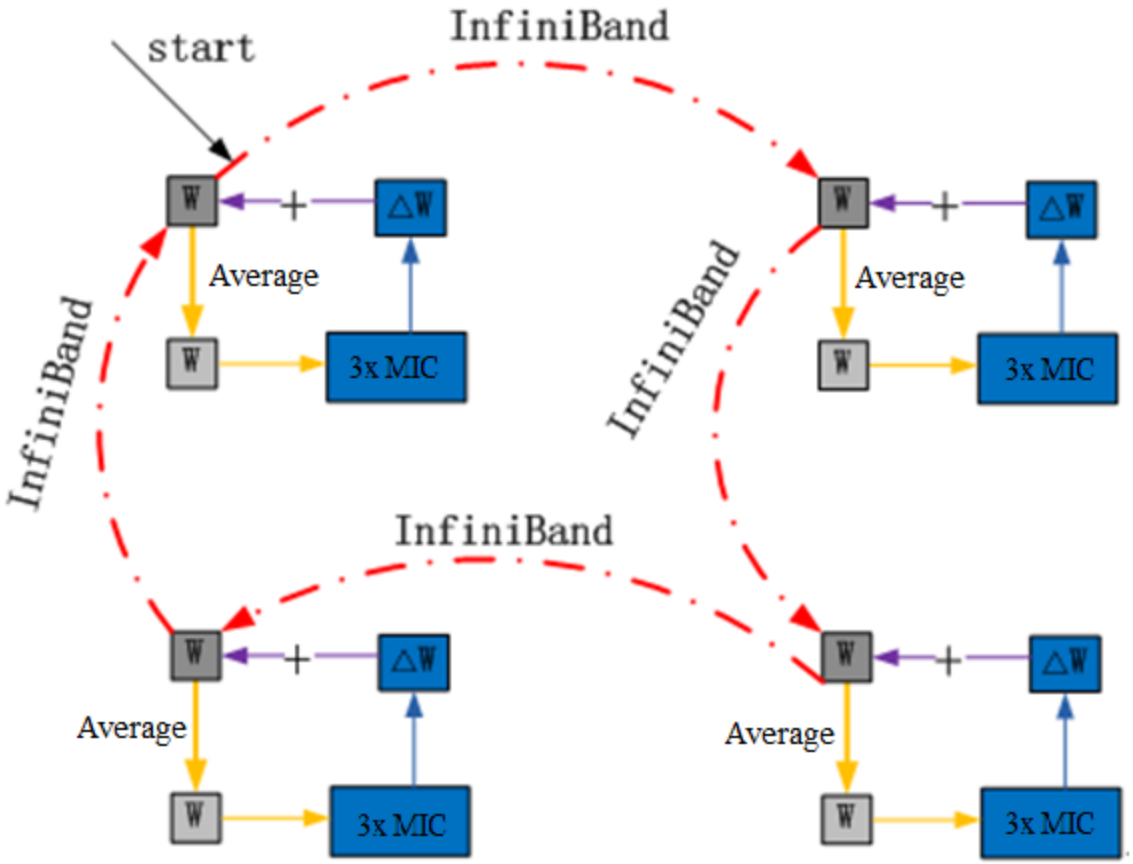
- 通过采用基于异步SGD的数据并行训练来实现多节点扩展：

	节点间	节点内
模型同步方案	环形同步	主从模式
数据并行方案	多个Batch并行	Batch内分割

- 在多节点间，进程按环形次序两两同步，减少通信开销
- 在单节点内，计算部分由多块Xeon Phi完成，CPU负责多个模型的收集、合并与分发

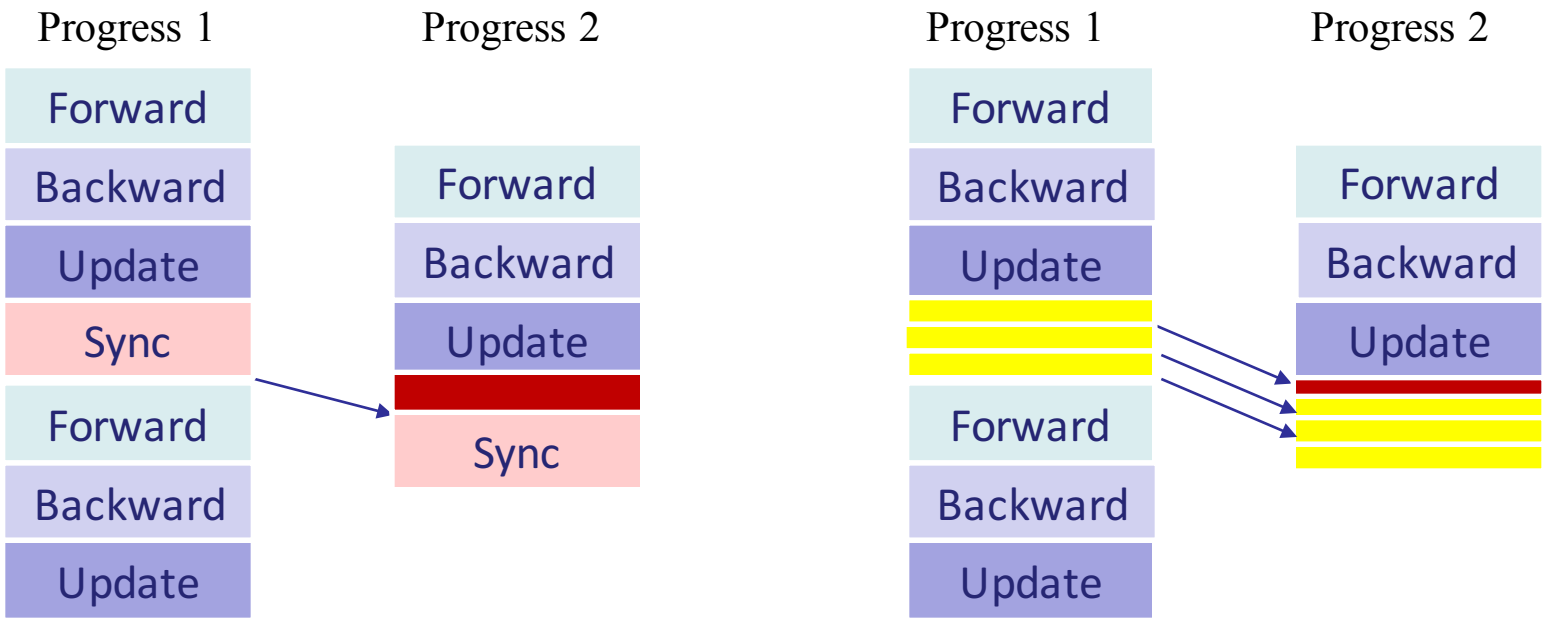
模型同步

- 使用环形通信结构
  - 每个训练进程在进行模型参数同步时，从前一进程中获取模型，做参数平均，再发送给后一进程



模型同步

- 使用环形通信结构
- 使用非阻塞发送，并分层同步模型，重叠模型平均与通信时间



Before Optimization

After Optimization

训练精度调优

- 异步并行训练会影响神经网络的参数收敛
- 调整相关参数，对程序的扩展性能与神经网络训练精度两者进行平衡：
  - 模型同步频率：

    降低频率，可增加并行扩展性，精度降低
  - 训练数据Batch的大小：

    增大Batch，充分利用MIC计算性能，精度降低
  - 学习速率：不影响程序性能，

    可以提高精度，也可能导致参数发散

测试结果

- 测试集群：
  - 天河二号：

    CPU：Intel Xeon E5-2692, x2 per node
    12 cores @ 2.200Ghz

    MIC：Intel Xeon Phi 31S1P, x3 per node

    Memory：64GB

    Interconnect: TH Express-2, 6.36GB/s

- 测试算例

算例内容	样本量	模型结构	精度要求
A 英语	10w	429, 2048, 8991	34.8%
B 普通话	5.1w	429, 2048, 9004	45.3%
C 四川方言	2.6w	429, 2048, 9004	43.1%

测试结果

- 在保证精度不变时的性能：

算例	节点数	节点间/内同步频率	学习速率	批大小	时间 (s)	实际精度
A(英语)	4	每 4 / 2 次迭代同步	0.768	672	11551.9	35.07%
B(普通话)	8	每 8 / 8 次迭代同步	2.7	1344	1179.3	45.61%
C(四川话)	8	每 8 / 4 次迭代同步	3.88608	1024	824.5	43.21%

- Notes：
  - 算例A：典型的大规模样本，相应的训练迭代次数很多，难以通过调整学习速率来达到预期结果
  - 算例C：存在一簇特殊的训练样本和验证样本，这与方言本身的特殊性有关

测试结果

- 保证扩展性的情形：
  - 均运行在8个节点上，使用24块Xeon Phi
  - 节点间进程每8次迭代进行同步
  - 节点内Xeon Phi卡每8次迭代进行同步

算例内容	样本量	学习速率	Batch大小	时间 (s)	理论精度	实际精度
英语	10w	0.768	1024	3551.9	34.8%	29.3%
普通话	5.1w	2.7	1344	1179.3	45.3%	45.6%
四川方言	2.6w	3.88608	1024	683.4	43.1%	41.8%

  - 精度的下降程度较小，且扩展性良好

测试结果

整体性能：

- 基于8节点、共24块MIC协处理器，加速约90~96倍
- 在8节点上的扩展性约为7.1倍
- 在24块MIC卡上的扩展性为最高至16~17倍

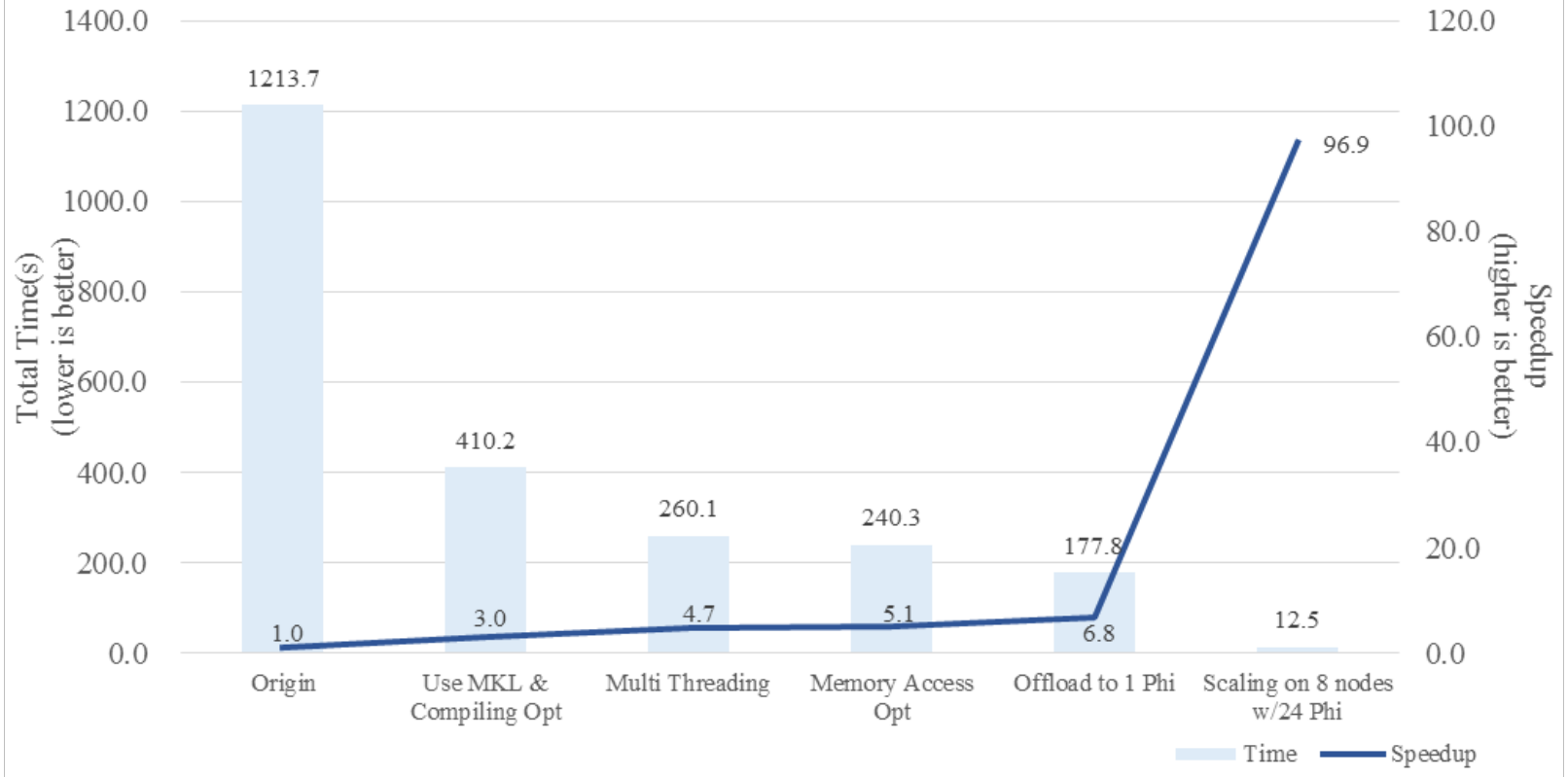


Fig. Performance Improvement Comparison

大纲

- 背景介绍
- 单节点优化
- 集群并行优化
- 总结

总结

- 在单节点上，通过编译优化、多线程、向量化以及访存优化等手段，可以获得约7倍的加速比
- 扩展到多个节点及多块加速卡上时，需要平衡模型的精度与程序的扩展性
- 并行训练时，在8个节点（24块Xeon Phi）上获得90~96倍的加速比，同时保证精度下降较少