ParSecureML: An Efficient Parallel Secure Machine Learning Framework on GPUs 论文复现

郑钧洛

摘要

机器学习在我们的日常生活中已经很常见，大量的数据不断产生并被传输到云端进行模型训练和数据处理，这就衍生出了一个问题：如何保护数据的安全。最近，有人提出了一种名为 SecureML 的安全机器学习系统，使用两方计算来解决这个问题，但是由于两方计算带来了额外的计算开销，所以安全机器学习比原来的机器学习方法慢了大约 2 倍。之前关于安全机器学习的工作主要集中在新颖的协议或提高准确率上，性能的提高却被忽视了，本文所复现的论文提出了一个基于 GPU 的框架 ParSecureML，可以提高基于两方计算的安全机器学习算法的性能。开发 ParSecureML 的主要挑战有三点，第一是复杂的计算模式，第二是 CPU 和 GPU 之间频繁的节点内数据传输，第三是复杂的节点间数据依赖。为了解决这些挑战，ParSecureML 带来了三种新颖的技术，分别是基于分析的自适应 GPU 引擎、节点内 CPU-GPU 细粒度协同的双管道设计和基于压缩的数据传输方法。根据复现结果显示，与最先进的框架相比，ParSecureML 实现了平均 18.8 倍的加速。

关键词：安全；GPU；机器学习；两方计算

1 引言

现如今，大规模机器学习任务通常在高性能计算服务器上执行，这些服务器大多由第三方提供，所以在此机器学习过程中的数据安全问题越来越受到重视。多方计算是一种保护数据安全的重要方法，被广泛运用在不同的应用场景中[1]。最近，多方计算被运用在机器学习中[2]，不同于差分隐私方法，多方计算将原始数据集分割成多个包含部分数据的加密副本，然后将这些加密副本分发到不同的服务器上，通过在多台服务器上处理加密数据，即使这些服务器是不可信的，多方计算也能提供高水平的数据安全保护。

两方计算，是多方计算的一种特殊形式，虽然复杂度较低，但是性能仍然较差，原因如下：第一，相比传统的、没有加密处理的机器学习算法，两方计算引入了大量的额外计算量；第二，在两方计算中，两台服务器之间需要互相通信，这会产生通信开销；第三，在两方计算中，不同计算步骤之间存在依赖关系，这限制了并行性。

近年来，出现了许多基于两方计算的安全机器学习应用，例如线性和逻辑回归、神经网络和 k 近邻算法。Mohassel 等人提出 SecureML[3]，它是目前最先进的基于双方计算的机器学习框架，SecureML 包含一套新颖的协议和模型，可以用于各种机器学习应用，但是 SecureML 比原始机器学习方法大约慢两倍。近来，GPU 被广泛用于加速机器学习[4]，然而目前并不存在 GPU 加速安全机器学习方面的研究，鉴于此，本文所复现的论文提出 ParSecureML，它是一个基于 GPU 的、高效的并行安全机器学习框架，可用于安全处理机器学习任务。ParSecureML 包含了三个新颖的技术，第一，开发了一个基于分析的自适应 GPU 引擎，通过分析两方计算过程，确定计算量最大的部分，并通过自适应地应

用 GPU 来加速这部分；第二，提出了一种节点内 CPU-GPU 细粒度协同的双管道设计，不仅可以实现 GPU 计算和 PCIe 数据传输的重叠，还可以实现不同神经网络层之间的潜在步骤重叠；第三，为了加速两台服务器之间的数据通信，提出了一种新颖的基于压缩的数据传输方法。

2 相关工作

近年来，有许多工作研究把 GPU 应用在高性能计算[5-11]和机器学习[12-15]方面。与 ParSecureML 相近的工作是 SecureML[3]，它把两方计算应用在机器学习上。与 SecureML 相比，ParSecureML 将两方计算的实用性提高到前所未有的水平。

两方计算是多方计算的代表[16]，多方计算将数据分割到多个服务器上，每个服务器只包含加密的部分数据，以防止数据泄露，不幸的是，参与方越多，性能下降就越明显，这是由于计算和通信开销增加导致的，所以很多研究人员都尝试去提高多方计算的效率[2,17-19]，Bogdanov 等人[18]研究并设计了高效的多方计算协议；Agrawal 等人[17]开发了一种定制的安全两方协议，用于 DNNs 的离散化训练；Mohassel 等人[3]在 SecureML 中使用两方计算作为基本算法。然而，GPU 作为机器学习中最广泛使用的加速器，还没有被用于两方计算的性能加速，本文所复现的论文首创使用 GPU 加速基于两方计算的安全机器学习任务。

3 本文方法

3.1 本文方法概述

ParSecureML 包含三个新颖的技术，分别是基于分析的自适应 GPU 引擎、用于重叠节点内数据传输和计算的双管道设计、和实现节点间通信的压缩传输设计。

如图 1所示，为基于两方计算的安全机器学习细节，其中机器学习方法选择 MLP 神经网络，数据集选择 MNIST，我们可以观察到离线阶段的“data distribution”步骤和在线阶段的最后一个计算步骤花费的时间最多，那么我们可以试着用 GPU 去加速这些步骤，这就是 ParSecureML 包含的第一个新颖技术，即基于分析的自适应 GPU 引擎。

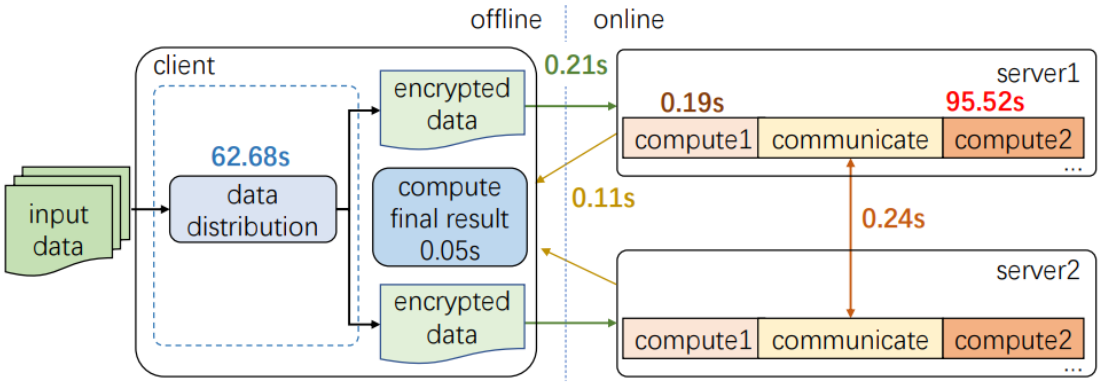

图 1: 两方计算的时间分解

从图 1中可以观察到，“compute1”和“communicate”步骤的用时都很短，所以不适合用 GPU 加速，而是适合用 CPU 进行处理，于是可以把“compute1”和“communicate”合为一个步骤，用 CPU 处理，把“compute2”交给 GPU 处理，由此可以设计第一个流水线用于重叠计算和 PCIe 通信。此外，通常机器学习任务的每层都包含几个步骤，一些不同层的步骤可以重叠执行，于是可以设计出第二个

流水线用于执行这些步骤。这就是 ParSecureML 包含的第二个新颖技术，即用于重叠节点内数据传输和计算的双管道设计。

对于节点间的通信，ParSecureML 在传输前对数据进行压缩，增加了数据的传输密度，从而提高了传输效率，这就是 ParSecureML 包含的第三个新颖技术，即节点间通信的压缩传输设计。

如图 2所示，为 ParSecureML 的总览图。首先在客户端使用 GPU 生成加密数据，并将加密数据传输到 server1 和 server2，然后，服务器在接收到数据后开始进行计算。“reconstruct”阶段包含服务器间通信的压缩，“GPU operation”阶段包含了主要的计算量、占用了大部分时间。值得注意的是，目前的机器学习任务通常包含一些层，在每一层中，都有前向传播和反向传播，前向传播和反向传播都有“reconstruct”和“GPU operation”两个阶段，ParSecureML 开发了不同层之间的流水线操作，以隐藏“reconstruct”阶段的通信开销。最后，服务器会将结果返回到客户端。

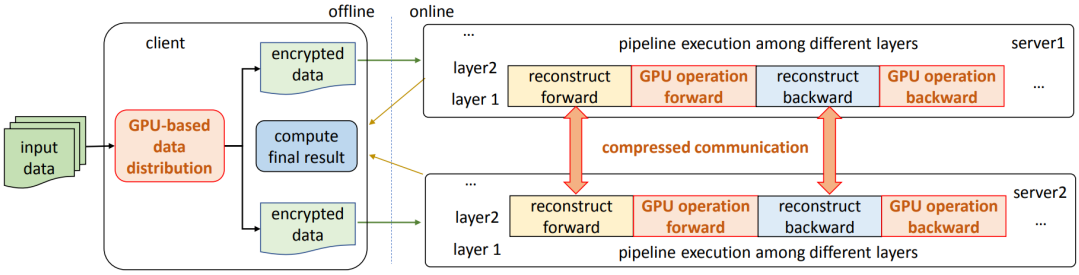

图 2: ParSecureML 总览图

3.2 分析引导的自适应 GPU 使用

由于两方计算包括离线和在线两部分，所以 GPU 加速也可以分为离线和在线两部分。

对于离线部分，目标是为两台服务器准备加密数据，如图 3所示，用户输入矩阵 A 和矩阵 B，然后离线部分对这两个矩阵进行各种处理，其中把矩阵 A 分成矩阵 A0、A1 和把矩阵 B 分成矩阵 B0、B1 这一步只需耗费很短时间，而矩阵 U 和 V 相乘得到矩阵 Z 这一步需要耗费很长时间，占离线部分总耗时超过 90%，这一步由密集的矩阵乘法计算组成，可以使用 GPU 加速。

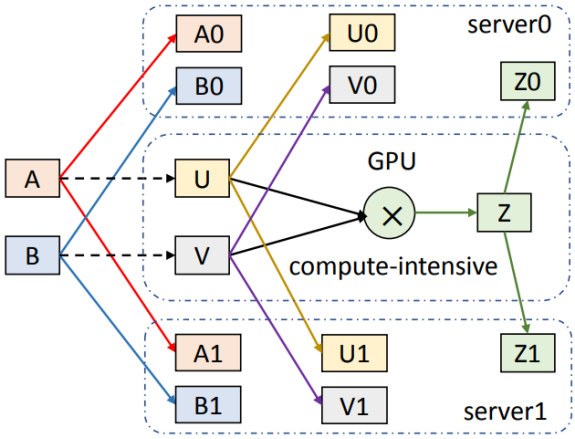

图 3: 离线部分数据处理示意图

对于在线部分，跟在线部分相关的计算是式 (2)、(3)，其中式 (2) 的计算量较小，式 (3) 的计算量较大，所以用 CPU 处理式 (2)、用 GPU 处理式 (3)。因为要用 GPU 处理式 (3)，所以需要把矩阵 E，F，A_i，B_i，和 Z_i 送到 GPU 中，式 (3) 可以表达为式 (4)，进一步地，式 (4) 可以表达为式 (5)，这样可以用一个加法操作代替一个乘法操作，从而减少开销。

$$E_i = A_i - U_i, F_i = B_i - V_i \tag{1}$$

$$E = E_0 + E_1, F = F_0 + F_1 \tag{2}$$

$$C_i = (-i) \times E \times F + A_i \times F + E \times B_i + Z_i \tag{3}$$

$$\begin{bmatrix} (-i) \times E & A_i & E \end{bmatrix} \times \begin{bmatrix} F \\ F \\ B_i \end{bmatrix} + Z_i, i \in 0, 1 \tag{4}$$

$$\begin{bmatrix} (-i) \times E + A_i & E \end{bmatrix} \times \begin{bmatrix} F \\ B_i \end{bmatrix} + Z_i, i \in 0, 1 \tag{5}$$

3.3 节点内 CPU-GPU 细粒度协同的双流水线设计

流水线用于重叠计算和通信，隐藏 PCIe 数据传输开销，经过分析了一系列的机器学习任务，可以发现典型的机器学习任务，尤其是深度学习，有两个特点：1）通常有几个相互依赖的神经网络层，2）机器学习任务包括前向传播和反向传播。神经网络层数由用户根据经验指定，通常大于两层以捕获非线性关系。基于这些分析，提出了一个双流水线设计，如图 4所示，第一个流水线用于重叠式 (5) 中的 GPU 计算和 PCIe 数据传输。

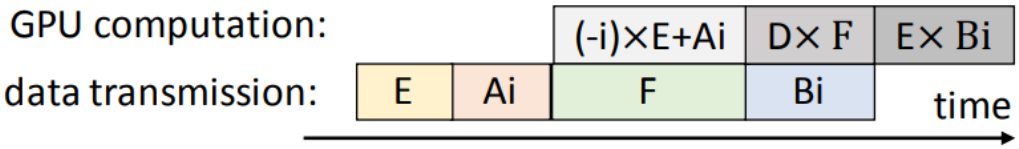

图 4: 用于重叠 GPU 计算和 PCIe 数据传输的流水线，D 为“$(-i) \times E + A_i$”的结果

第二个流水线用于重叠不同层的一些步骤，通过观察发现后续层的处理依赖于当前层的前向传播，所以下一层的前向传播“reconstruct”步骤不能与当前层的前向传播“GPU operation”步骤重叠。但是反向传播中的“reconstruct”步骤不需要等待后续的层，因此可以在流水线中与下一层的传播一起进行。如图 5所示，对于反向传播“reconstruct”步骤，部分计算依赖于同一层的正向传播“GPU operation”步骤，因此可以把这些计算保留在反向传播“reconstruct”步骤中，另一部分计算依赖于下一层的结果，因此将计算分配给反向传播“GPU operation”步骤。所以当前层的正向传播“reconstruct”步骤可以与上一层的反向传播“reconstruct”步骤重叠。

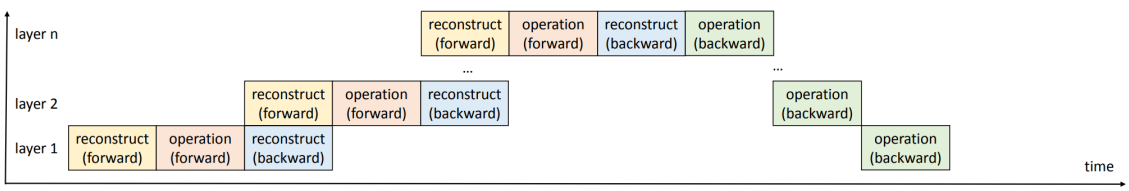

图 5: 第二个流水线设计

3.4 节点间通信压缩传输

为了计算 E 和 F，服务器之间需要通信 E_i 和 F_i，而 E_i 和 F_i 依赖于 A_i 和 B_i，在机器学习任务中，A 和 B 可以是模型参数，它们的迭代矩阵通常是稀疏的。用 $A_{i,j}$ 和 $B_{i,j}$ 表示 A 和 B 在服务器 i 的第 j 次迭代，所以 $A_{i,j+1}$ 和 $B_{i,j+1}$ 可以表达为式 (6)，其中 $\Delta_{ij}^{A/B}$ 代表两次迭代之间的变化，即梯度。

$$A_{i,j+1} = A_{i,j} + \Delta_{ij}^{A}, B_{i,j+1} = B_{i,j} + \Delta_{ij}^{B} \tag{6}$$

得到 $A_{i,j}$ 和 $B_{i,j}$ 的表达式后，我们可以得到 $E_{i,j}$ 和 $F_{i,j}$ 的表达式如式 (7)、式 (8) 所示。所以，如果我们可以提前判断 $\Delta_{ij}^{A/B}$ 是稀疏的，那么我们只需在两台服务器之间传输变化值 $\Delta_{ij}^{A/B}$。

在传输 E_i 和 F_i 之前，我们先检查 Δ_{ij}^{A} 和 Δ_{ij}^{B}，如果 Δ_{ij}^{A} 和 Δ_{ij}^{B} 是稀疏的（默认设置为矩阵中 75% 的元素为 0），则使用压缩稀疏行格式（CSR）[20]来存储它，然后传输压缩后的 Δ_{ij}^{A} 或 Δ_{ij}^{B}，如果不是稀疏的，则传输原始数据。

$$E_{i,j+1} = A_{i,j+1} - U_i = A_{ij} + \Delta_{ij}^{A} - U_i = E_{ij} + \Delta_{ij}^{A} \tag{7}$$

$$F_{i,j+1} = B_{i,j+1} - V_i = B_{ij} + \Delta_{ij}^{B} - V_i = F_{ij} + \Delta_{ij}^{B} \tag{8}$$

4 复现细节

4.1 与已有开源代码对比

本文的复现实验参考了原论文提供的开源代码，并在此基础上进行了改进、增加了新功能。原来的程序在执行时，用户如果想要更改数据集，就需要进入到源码中，找到选择数据集的代码部分，然后对此部分代码进行更改，十分麻烦，所以本文对此进行改进，设计出一个更为友善的人机交互界面，使用户在程序开始运行的时候，可以对数据集类型进行选择。

4.2 实验环境搭建

基线方法选择 SecureML；衡量性能的方法是选用四个典型的机器学习任务（CNN，MLP，Linear regression 和 Logistic regression）去评估 ParSecureML 和 SecureML 的性能；数据集选用 VGGFace2，NIST，SYNTHETIC，和 MNIST；平台：本次实验使用一台服务器模拟三个节点，这台服务器装有三个 Intel(R) Xeon(R) CPU E5-2650 v4 @ 2.20GHz CPU 和一个 Nvidia Tesla V100 GPU，操作系统为 Ubuntu 18.04.5。

4.3 界面分析与使用说明

如图 6所示，为本实验的实验界面，开始执行程序后，会让用户选择数据集，用户输入相应的数据集序号，然后按“回车”键确认，接下来，程序便会根据用户所选的数据集，执行不同的机器学习任务，并列出不同机器学习任务所耗费的总时间、离线阶段时间、和在线阶段时间，时间单位为秒。

```
Please enter a number to choose dataset:1.VGGFace2  2.NIST  3.SYNTHETIC  4.MNIST
4
+-------------------------------------------------------------+
| Application: Single matrix multiplication                   |
|-------------------------------------------------------------|
| All Time       | Offline Phase Time    | Online Phase Time  |
| 1.63           | 1.21                  | 0.41               |
+-------------------------------------------------------------+
+-------------------------------------------------------------+
| Application: Linear regression                              |
|-------------------------------------------------------------|
| All Time       | Offline Phase Time    | Online Phase Time  |
| 2.64           | 1.51                  | 1.13               |
+-------------------------------------------------------------+
+-------------------------------------------------------------+
| Application: Logistic regression                            |
|-------------------------------------------------------------|
| All Time       | Offline Phase Time    | Online Phase Time  |
| 2.62           | 1.51                  | 1.11               |
+-------------------------------------------------------------+
+-------------------------------------------------------------+
| Application: MLP                                            |
|-------------------------------------------------------------|
| All Time       | Offline Phase Time    | Online Phase Time  |
| 5.49           | 1.54                  | 3.94               |
+-------------------------------------------------------------+
+-------------------------------------------------------------+
| Application: CNN                                            |
|-------------------------------------------------------------|
| All Time       | Offline Phase Time    | Online Phase Time  |
| 7.99           | 1.56                  | 6.42               |
+-------------------------------------------------------------+
```

图 6: 实验界面

4.4 创新点

本文在原论文开源代码的基础上，改进了人机交互界面，使得用户可以在程序开始执行时选择想要运行的数据集，而不用每次都进入源码，找到选择数据集的代码部分，再对此部分代码进行更改，从而提高了便捷度、提高了实验调试效率。

5 实验结果分析

如图 7所示，为本文复现实验的结果，在全局性能上，ParSecureML 平均比 SecureML 快了 18.8 倍。

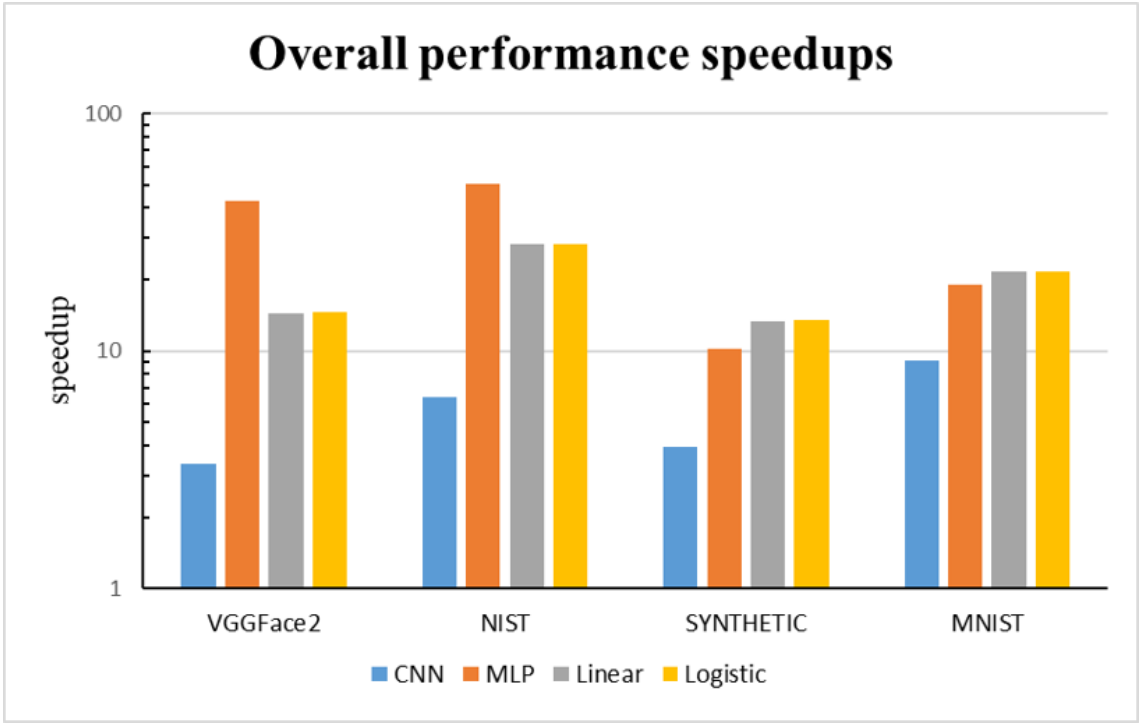

图 7: 全局性能加速

如图 8所示，为在线阶段的性能提升情况，ParSecureML 的在线阶段平均比 SecureML 的在线阶段快了 31.4 倍，这表明了全局性能的提升主要来自于在线阶段性能的提升。

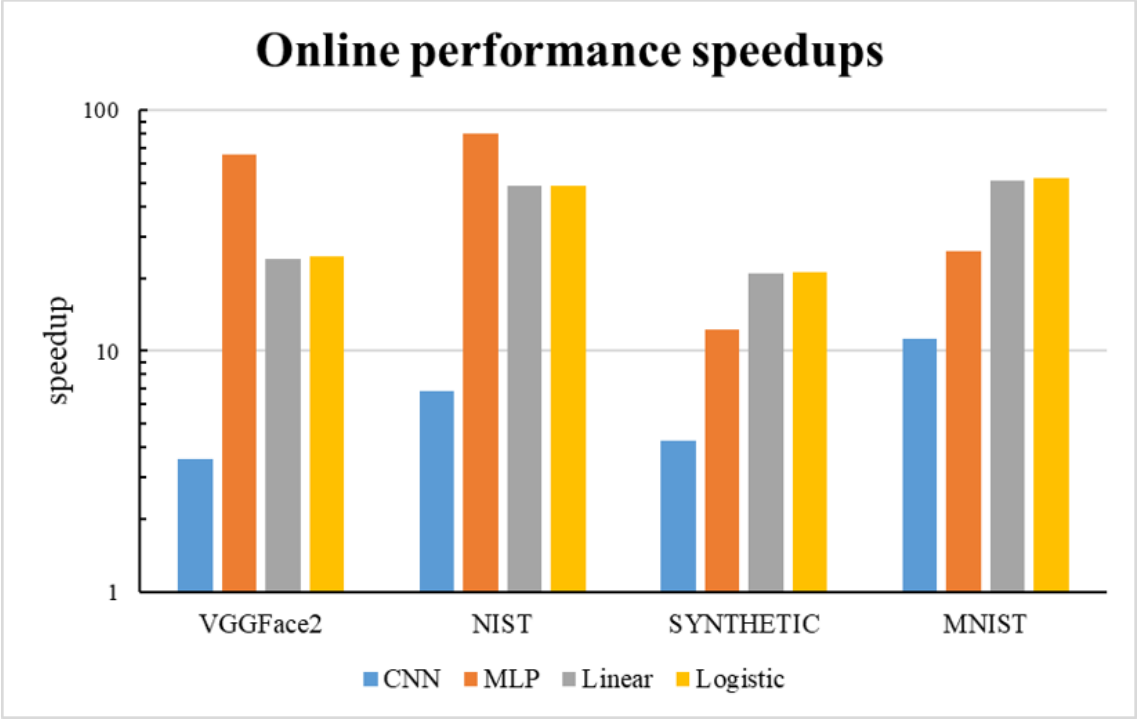

图 8: 在线阶段性能加速

如图 9所示，为离线阶段的性能提升情况，ParSecureML 的离线阶段平均比 SecureML 的离线阶段快了 1.3 倍，提升较少，原因是离线部分的计算量较少，性能提升的空间不大，而且由于离线阶段耗时占总耗时的比例较小，所以离线阶段不会影响到总体性能的提升。

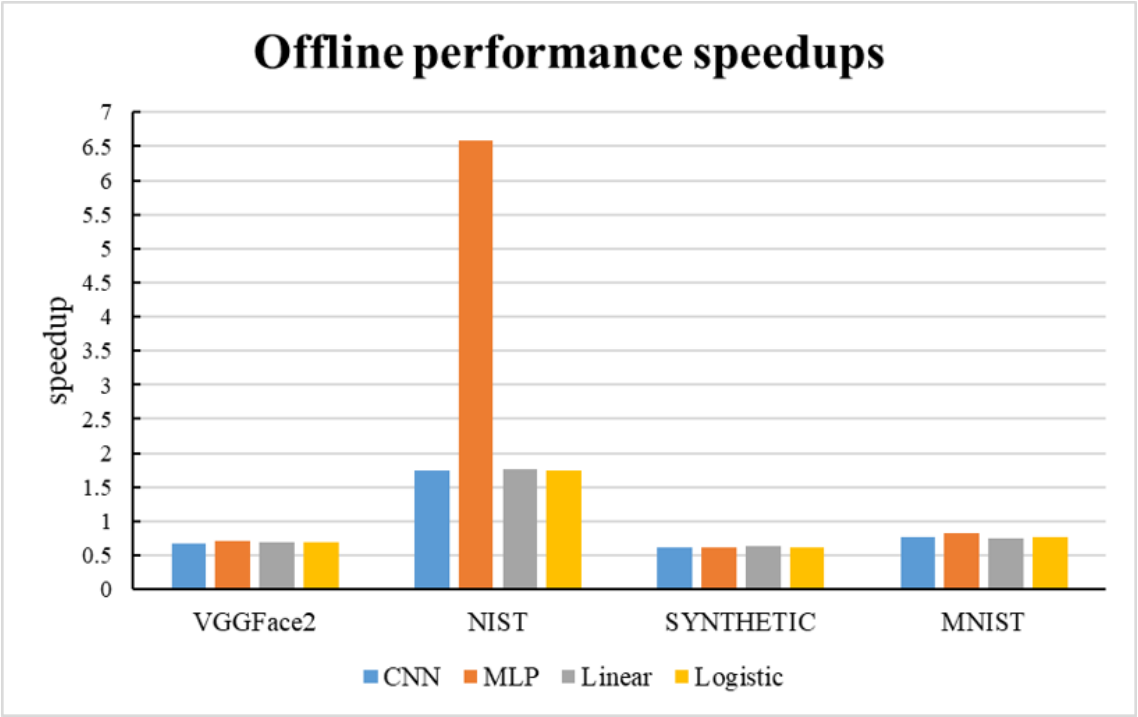

图 9: 离线阶段性能加速

6 总结与展望

ParSecureML 提出了三个新颖的技术以提高基于两方计算的机器学习任务的性能：1）基于分析的自适应 GPU 加速、2）用于重叠节点内数据传输和计算的双流水线设计、和 3）实现节点间通信的

压缩传输设计，使得在全局性能上，ParSecureML 平均比 SecureML 快了 18.8 倍，这说明了用 GPU 加速安全机器学习是很有潜力的，未来可以把 GPU 加速拓展到更多的安全机器学习算法中。

参考文献

[1] ARCHER D W, DAN B, YEHUDA L, et al. From Keys to Databases—Real-World Applications of Secure Multi-Party Computation[J]. The Computer Journal, 2018(12): 1749-1771.

[2] HONG C, HUANG Z, LU W J, et al. Privacy-preserving collaborative machine learning on genomic data using TensorFlow[Z]. 2020.

[3] MOHASSEL P, ZHANG Y. SecureML: A System for Scalable Privacy-Preserving Machine Learning [C]//2017 IEEE Symposium on Security and Privacy (SP). 2017: 19-38. DOI: 10.1109/SP.2017.12.

[4] UPADHYAYA S R. Parallel approaches to machine learning—A comprehensive survey[J]. Journal of Parallel & Distributed Computing, 2013, 73(3): 284-292.

[5] LI C, SONG S L, DAI H, et al. Locality-Driven Dynamic GPU Cache Bypassing[C]//Acm on International Conference on Supercomputing. 2015.

[6] MATAM K K, KOTHAPALLI K. Accelerating Sparse Matrix Vector Multiplication in Iterative Methods Using GPU[C]//IEEE. 2011.

[7] ODEN L, FRÖNING H, PFREUNDT F J. Infiniband-Verbs on GPU: A Case Study of Controlling an Infiniband Network Device from the GPU[C]//, Fourth International Workshop on Accelerators and Hybrid Exascale Systems (AsHES), in conjunction with IPDPS 2014. 2014.

[8] OHSHIMA S, YAMAZAKI I, IDA A, et al. Optimization of Numerous Small Dense-Matrix–Vector Multiplications in H-Matrix Arithmetic on GPU[C]//2019 IEEE 13th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC). 2019.

[9] WU G, GREATHOUSE J L, LYASHEVSKY A, et al. GPGPU performance and power estimation using machine learning[C]//2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA). 2015.

[10] YAMAZAKI I, ANZT H, TOMOV S, et al. Improving the performance of CA-GMRES on multicores with multiple GPUs[C]//2014 IEEE International Parallel & Distributed Processing Symposium (IPDPS). 2014.

[11] ZHANG F, LIN H, ZHAI J, et al. An adaptive breadth-first search algorithm on integrated architectures [J]. The Journal of Supercomputing, 2018, 74: 6135-6155.

[12] ISLAM T Z, THIAGARAJAN J J, BHATELE A, et al. A Machine Learning Framework for Performance Coverage Analysis of Proxy Applications[C]//SC16: International Conference for High Performance Computing, Networking, Storage and Analysis. 2016.

[13] Eunjung, Park, John, et al. Predictive Modeling in a Polyhedral Optimization Space[J]. International Journal of Parallel Programming, 2013.

[14] TANG S, HE B, YU C, et al. A Survey on Spark Ecosystem: Big Data Processing Infrastructure, Machine Learning, and Applications[J]. IEEE Transactions on Knowledge and Data Engineering, 2022(34-1).

[15] XIE C, FU X, CHEN M, et al. OO-VR: NUMA Friendly Object-Oriented VR Rendering Framework For Future NUMA-Based Multi-GPU Systems[J]., 2020.

[16] GOLDREICH O. Secure Multi-Party Computation[J]. Chapman & Hall/CRC,

[17] AGRAWAL N, SHAMSABADI A S, KUSNER M J, et al. QUOTIENT: Two-Party Secure Neural Network Training and Prediction[J]. ACM, 2019.

[18] BOGDANOV D, NIITSOO M, TOFT T, et al. High-performance secure multi-party computation for data mining applications[J]. International Journal of Information Security, 2012.

[19] HIRT M, MAURER U, PRZYDATEK B. Efficient Secure Multi-party Computation[J]. Springer-Verlag, 2000.

[20] BELL N, GARLAND M. Efficient Sparse Matrix-Vector Multiplication on CUDA[J]., 2009.