第1卷 第1期 2019年10月
Vol.1 No.1 Oct.2019

ISSN 2096-2223
CN 10-1649/TP

文献DOI：
10.11871/
文献PID：
21.86101.2/

文献二维码：

页码：

Angel$^+$：基于Angel的分布式机器学习平台

张智鹏[1]，江佳伟[2]，余乐乐[2]，崔斌[1]

1．北京大学，计算机科学与技术系，高可信软件技术教育部重点实验室，北京100871
2．腾讯公司，北京100193

摘　要：【目的】随着大数据时代的来临，数据变得高维、稀疏，机器学习模型也变得复杂、高维，因此也给分布式机器学习系统带来了很多挑战。尽管研究人员已经开发了很多高性能的机器学习系统，比如TensorFlow、 PyTorch、XGBoost等，但是这些系统存在以下两个问题：（1）不能与现有的大数据系统很好的结合；（2）不够通用，这些系统往往是为了某一类机器学习算法设计。【方法】为了解决以上两个挑战，本文介绍Angel$^+$：一个基于参数服务器架构的分布式机器学习平台。【结果】Angel$^+$能够高效的支持现有的大数据系统以及机器学习系统——依赖于参数服务器处理高维模型的能力，Angel$^+$能够以无侵入的方式为大数据系统（比如Apache Spark）提供高效训练超大机器学习模型的能力，并且高效的运行已有的分布式机器学习系统（比如PyTorch）。此外，针对分布式机器学习中通信开销大和掉队者问题，Angel$^+$ 也提供了模型平均、梯度压缩和异构感知的随机梯度下降解法等。【结论】笔者结合Angel$^+$开发了很多高效、易用的机器学习模型，并且通过实验验证了Angel$^+$平台的高效性。

关键词：分布式机器学习平台；参数服务器；大数据处理系统；分布式机器学习系统

Angel$^+$: A Large-Scale Machine Learning Platform on Angel

Zhang Zhipeng[1], Jiang Jiawei [2], Yu Lele [2], Cui Bin [1]

1. Department of Computer Science & Key Laboratory of High Confidence Software Technologies (MOE), Peking University, Beijing 100871, China
2. Tencent, Beijing 100193, China

Abstract: **[Objective]** Real-world data becomes much more complex, sparse and high-dimensional for the big data shock in this era. According to this, modern ML models are designed in a deep, complicated way, which arises challenges when designing a distributed machine learning (ML) system. Though researchers have developed many efficient centralized ML systems like TensorFlow, PyTorch and XGBoost, these systems suffer from the following two problems: (1) They cannot integrate well with existing big data systems, (2) they are not general enough and are usually designed for specific ML models. **[Methods]** To tackle these challenges, we introduce Angel$^+$, a large-scale ML platform based on parameter servers. **[Results]** With the power of parameter servers, Angel+ can efficiently support existing big data systems and ML systems without neither breaking the core of big data systems, Apache Spark for instance, nor degrades the computation efficiency of current ML frameworks like PyTorch. Furthermore, Angel$^+$ provides algorithms like model averaging, gradient compression and heterogeneous-aware stochastic gradient descent, to deal with the huge communication cost and the straggler problem in distributed training process. **[Conclusions]** We also enhance the usability of Angel$^+$ by providing efficient implementation for many ML models. We conduct extensive experiments to demonstrate the superiority of Angel$^+$.

Keywords: machine learning platform; parameter servers; big data systems; distributed machine learning systems

国家基金：国家重点研发计划重点专项(2018YFB1004403)；国家自然科学基金(61832001)

引言

近年来，随着互联网、通信、电子商务等行业的不断发展，各种类型的数据呈爆炸式增长，大数据已经成为研究界和工业界关注的热点。大数据中往往隐含着小数据量不具备的深度知识和潜在价值，挖掘出这些信息能够提升行业和企业的管理能力、决策水平和经济效益，为社会和个人提供更多智能和便利的服务。大数据分析中，机器学习技术是被广泛应用的工具，在单机上串行或者并行处理的机器学习算法已经有了比较成熟的研究。

但是，当今动辄达到数百 TB 甚至 PB 规模的数据以及应用的复杂程度已经远远超过了单机系统的计算和存储能力。数据量的增加通常伴随着更高维的机器学习模型，更多的模型参数以及更为复杂的训练过程。在业界真实的场景中，一个大规模机器学习系统需要有能够对 PB 量级的训练数据和数百亿维度的模型参数进行训练的能力。

1 分布式机器学习的挑战和解决方法

为了方便大规模机器学习任务的开发和应用，研究人员设计了各种各样的系统来解决大规模机器学习的问题。在大数据的背景下，分布式机器学习系统作为机器学习和分布式系统的交叉领域，有着独特的挑战：

(1) 高维模型。在业界真实的场景中，数据特征的维度往往能够达到十亿甚至百亿的规模。而在训练高维机器学习模型的过程中，各个计算节点需要互相通信以完成分布式的训练。这个过程往往意味着巨大的存储和通信开销，远远超过了单个机器的能力。

(2) 与现有大数据系统的兼容性。训练机器学习模型通常需要大量的训练数据，而这些数据通常由大数据系统（比如 Apache Spark[11]）通过处理分析得到。现有的机器学习系统都忽略了这一模块，不能很好的与大数据系统兼容，导致了大数据系统与机器学习系统之间昂贵的数据移动的开销。

(3) 通用性。研究人员针对不同模型的特点开发了各种不同的机器学习模型，比如 TensorFlow[1]，PyTorch，MXNet[2] 等专注于深度学习模型，Petuum[3] 专注于浅层模型，而 XGBoost[4] 专注于梯度提升树模型。这导致了人们在使用不同机器学习模型的时候，需要学习并维护新的机器学习系统，有较高的学习和系统维护的成本。

为了解决以上三个挑战，本文提出一个基于参数服务器的大规模机器学习平台 Angel+。Angel+ 的系统架构如图 1 所示，包含以下三个部分：

(1) 通用的参数服务器模块。为了解决高维模型在训练过程中存储和通信开销大的问题，Angel+ 使用参数服务器架构 [5,6] 来存储和管理模型。具体的，Angel+ 使用多个参数服务器来统一的存储高维机器学习模型。它将机器学习模型按照一定的逻辑切片到不同的服务器上，然后统一的向下层应用提供 pull/push 等接口，以完成计算节点与参数服务器节点之间的通信。此外，参数服务器模块也负责实现各种不同的同步协议，比如严格同步协议 BSP，异步协议 ASP 以及有限异步协议 SSP。

(2) Spark-on-Angel 模块。为了更好的兼容现有的大数据生态系统，Angel+ 在不改变 Spark 核心代码的情况下，以无入侵的方式实现了 Spark-on-Angel 模块，从而使得 Spark 用户能够以较低的成本享受到参数服务器高效的通信模块，并且避免了大数据系统与机器学习系统之间昂贵的数据移动。具体的，Angel+ 将参数服务器节点和 Spark 的计算节点分别启动为两个不同的进程：计算节点依赖 Spark RDD 强大的功能进行数据预处理、模型梯度的计算等；而参数服务器节点负责模型的存储、更新等。Spark 计算节点通过 PS-Client 与参数服务器节点进行高效的通信。

(3) PyTorch-on-Angel 模块。Angel+ 作为一个通用的机器学习平台，也能与现有的机器学习平台进行兼容，这里以 PyTorch 为例。在 PyTorch-on-Angel 模块中，我们以 native C++ 的方式将 PyTorch 运行

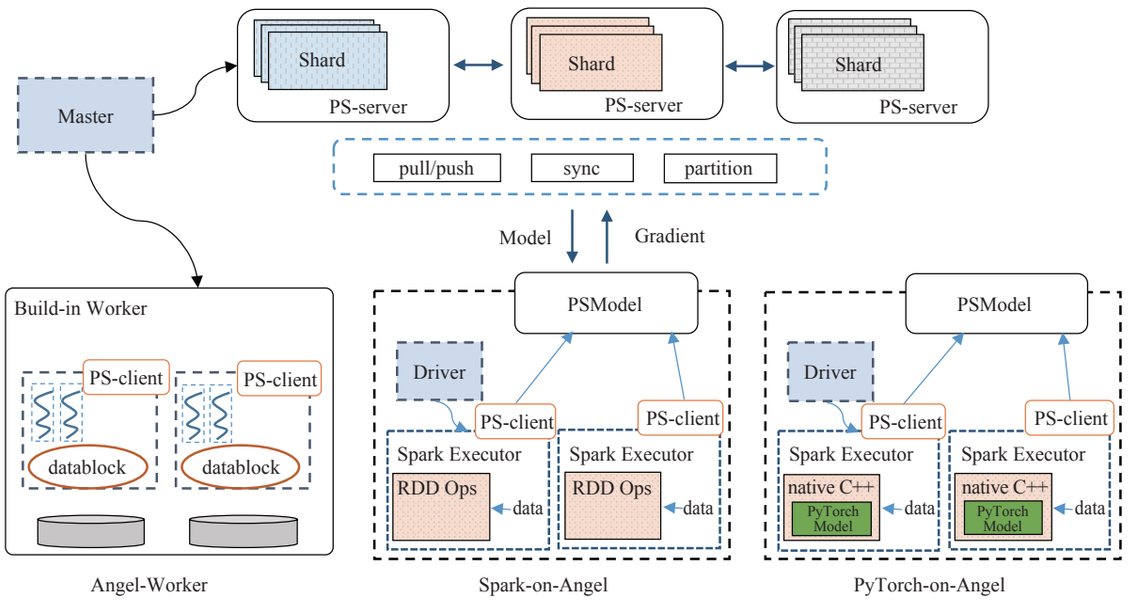


图 1　Angel+ 系统架构

Fig. 1 System architecture of Angel+

在 Angel$^+$ 中。如图 1 所示，用户以 native C++ 加载已经写好的 PyTorch 模型，然后使用 Spark-on-Angel 模块向 PyTorch 模块传入数据进行梯度的计算，并且使用参数服务器模块进行模型的存储和更新。

此外，Angel$^+$ 还从算法层面对分布式环境下的通信开销和掉队者问题分别做了优化：

Angel$^+$ 的通信开销。在分布式训练中，不同的计算节点之间要频繁的交换梯度 / 模型等信息来完成模型的训练。当数据维度较高（业界通常会达到十亿甚至百亿的级别），模型较大时，各个计算节点之前的通信代价变得不可忍受。为了减少计算节点和参数服务器节点之间的通信代价，Angel$^+$ 从减少单次通信量和减少通信次数两个角度进行了优化。为了减少通信频次，Angel$^+$ 提出了一种基于模型平均 [8] 的随机梯度下降训练方法，即通过多次局部更新再进行全局同步，进而大大减少全局通信的次数；为了减少单次通信量的大小，Angel$^+$ 基于数据草图提出了对稀疏梯度进行压缩的方法 SketchML[9].

Angel$^+$ 的掉队者问题。在每一轮迭代中，每个计算节点都要进行同步梯度 / 模型，因此较慢的机器会成为整个系统的运行瓶颈。为了解决掉队者问题，Angel$^+$ 基于参数服务器提出一种异构感知的分布式梯度下降训练方案 [10]。具体的，Angel$^+$ 通过允许节点之间异步的更新模型，减少了掉队者问题带来的影响。同时，由于掉队者对模型的更新通常是较为陈旧的，Angel$^+$ 进一步对掉队者的模型更新施加较大的惩罚项，进而保证了问题收敛的正确性。

笔者结合 Angel$^+$ 平台的特性，开发了很多高效、易用的机器学习算法，包括逻辑回归、支持向量机、矩阵分解、KMeans、梯度提升树、因子分解机、主题模型、图嵌入、图卷积神经网络等。本文用实验证明了 Angel$^+$ 平台的高效性。此外，Angel$^+$ 已经被部署在业界超过 2 年，运行着数以千计的真实业务。

内容组织：笔者先介绍 Angel$^+$ 的系统结构，包括基于 Angel 的参数服务器模块，Spark-on-Angel 模块以及 PyTorch-on-Angel 模块。接下来笔者将介绍 Angel$^+$ 分别就通信开销和掉队者问题上提出的优化算法，最后通过实验展示 Angel$^+$ 的高效性。

2 Angel+ 系统框架

Angel+ 基于分布式机器学习系统 Angel[5] 进行了进一步扩展，采用参数服务器架构来统一的管理机器学习模型，并在此基础上提供了一套机器学习训练模块 Angel-Worker。此外，Angel+ 还提供两个子模块 Spark-on-Angel 和 PyTorch-on-Angel，分别兼容了大数据处理系统 Apache Spark 和机器学习系统 PyTorch。接下来笔者将介绍各个模块的主要内容。

2.1 参数服务器模块

Angel+ 采用通用的参数服务器架构来统一的管理机器学习模型，即使用多个服务器来存储机器学习模型，并向各个子模块提供参数访问、参数更新的接口。接下来，笔者就模型划分策略、模型访问接口和模型同步策略一一介绍。

2.1.1 模型划分策略

在 Angel+ 中，我们将模型参数建模为一个二维矩阵。在此基础上，Angel+ 使用一个 PartitionKey 来描述每一个模型分区，包括该分区的起始行、终止行、起始列和终止列。本质来说，每个参数服务器维护的是整个模型的一个矩阵区域。在任务启动阶段，主节点会将模型进行划分，得到所有分区的 PartitionKey，然后将 PartitionKey 分配给不同的服务器节点，每个服务器节点得到 PartitionKey 以后可以随机初始化自身内容。

2.1.2 模型访问接口

为了方便计算节点的访问，Angel+ 提供了默认的 Pull/Push 接口。此外，为了支持服务器端的计算，Angel+ 还提供了大量的行访问 / 列访问的算子 [7]，如表 1 所示。

表 1 Angel+ 中参数服务器的操作算子

Table 1 PS operators in Angel+

操作类别	操作算子
行访问算子	pull, push, sum, nnz, norm2
列访问算子	axpy, dot, copy, sub, add, mul, div
创建模型算子	derive, dense, sparse

值得一提的是，利用这些高级算子使得用户能够在服务器端也执行计算，这样可以减少部分模型训练中的通信开销。此外，用户也可以自己定义算子，实现自己需要的执行逻辑。

2.1.3 模型同步策略

模型同步操作决定了执行任务中不同计算节点之间的同步效率，高效的同步操作能够保证计算节点可以尽早知道其他计算节点的状态，从而减少计算节点之间的等待时间。

Angel+ 支持同步执行协议 BSP，异步执行协议 ASP 和有限异步协议 SSP 三种同步模式，用户可以通过配置参数从而选择不同的同步方式。在 Angel+ 中，我们以模型分区粒度的向量时钟来实现精准的控制。

2.2 Angel-Worker 模块

Angel-Worker 模块是计算节点的一种实现方式，它不依赖于已有的大数据系统或者机器学习系统，提供了数据的读取、重划分，以及模型的训练等功能。Angel-Worker 作为计算节点，与主节点、参数服务器节点一起完成分布式机器学习的训练，它们的分工如下：

主节点 (master): 主节点负责申请资源，启动 / 杀死计算节点和服务器节点。同时，主节点也负责调度计算节点和参数服务器节点。

计算节点 (worker): 计算节点负责执行用户提交的任务逻辑。每个计算节点从 HDFS 读取数据，从服务器节点读取参数，进行模型的更新，并且负责将模型更新发送到参数服务器。

服务器节点 (server): 参数服务器节点负责分布式的存储模型参数。每个参数服务器节点负责存储一部分模型参数，向计算节点提供高效的访问接口，并且利用从计算节点得到的信息来更新模型。

接下来，笔者以逻辑回归为例介绍 Angel-Worker 模式的执行流程。

（1）任务提交。用户提交逻辑回归任务，并且制定整个任务的配置信息，包括计算节点、参数服

务器节点的个数以及资源配置情况；（2）参数服务器启动。主节点将模型参数划分为多个矩阵块，并做好从模型分区到参数服务器节点之间的映射关系，最后申请资源，启动参数服务器节点，并且初始化模型参数；（3）计算节点启动。待参数服务器节点启动后，Angel$^+$ 继续申请资源，启动计算节点进程；（4）加载训练数据。主节点根据数据的规模和计算节点的数目划分训练数据；然后每个计算节点分别加载各自的训练数据；（5）计算模型更新。每个计算节点去参数服务器节点拉取模型，进行梯度的计算，并且将模型更新推送到参数服务器；（6）更新模型。收集到各个节点的梯度后，参数服务器节点进行模型的更新；然后重复第 5 步和第 6 步直至收敛。（7）保存模型。服务器节点将模型存储至外存。

2.3 Spark-on-Angel 模块

在 Angel-Worker 模块中，训练数据往往是通过大数据系统比如 Apache Spark[11] 预处理产生并通过外存读取，此过程会导致较大的数据移动代价。为了减少分布式环境下数据移动的代价，Angel$^+$ 提供了 Spark-on-Angel 模块。

如图 1 所示，在 Spark-on-Angel 模块中，Angel$^+$ 利用 Spark 的计算节点来预处理数据以及计算模型更新，并且利用参数服务器来管理、更新模型参数。Angel$^+$ 将参数服务器节点和 Spark 计算节点启动为两个独立的应用，并且为 Spark 计算节点访问参数服务器节点提供了丰富、易用的的访问接口。

Spark-on-Angel 模块没有破坏 Spark 的核心代码，因此 Spark 用户能够以几乎透明的方式使用直接利用 Spark 进行高维机器学习模型的训练，从而避免了大数据系统和机器学习系统之间数据移动的代价。

2.4 PyTorch-on-Angel 模块

研究人员针对不同的机器学习模型开发了很多高效系统。比如 TensorFlow、PyTorch、MXNet 专注于深度学习模型，XGBoost 专注于梯度提升树模型，而 Petuum 专注于逻辑回归、矩阵分解、主题模型等传统机器学习模型。因此人们为了使用不同的机器学习模型通常需要使用不同的机器学习系统。为了减少使用不同机器学习系统的学习和系统维护成本，Angel$^+$ 也尝试支持这些系统在参数服务器上的运行，这里以 PyTorch 为例。

在 PyTorch-on-Angel 模块中，用户通过 python 编写 PyTorch 代码，然后通过 C++ native 的方式将序列化后的 PyTorch 模型加载到 Spark 计算节点中，从而无缝的使用运行在 Angel$^+$ 上，享受参数服务器带来的处理高维模型的能力。

3 通信开销

在分布式环境中，计算节点与参数服务器节点需要频繁的交换梯度 / 模型信息。当模型维度较高时，会带来较大的通信开销。本章首先介绍分布式随机梯度下降（SGD）的背景，然后介绍 Angel$^+$ 支持的两种通信优化算法。

Algorithm 1: Distributed SGD $\{T, \eta\ w, X, m\}$

Master:
for *Iteration* $t = 0$ **to** T **do**
    Issue WorkerTask(t) to workers;
    Issue ServerTask(t) to parameter servers;
Worker r = 1, ..., m:
Function *WorkerTask(t):*
    Get model w_{t-1} from the servers;
    Sample a batch of data X_{br} from X_r;
    Compute gradient $g_t^r \leftarrow \sum_{x_i \in X_{br}} \nabla l(w_{t-1}, x_i)$;
    Send gradient g_t^r to parameter servers;
Parameter Server:
Function *ServerTask(t):*
    Aggregate gradient as $g_t \leftarrow \sum_{r=1}^{m} g_t^r$;
    Update the model as
    $w_t \leftarrow w_{t-1} - \eta \cdot (g_t) - \eta \cdot \nabla\Omega(w_{t-1})$;

图 2　分布式 SGD 算法实现

Fig. 2 Implementation of distributed SGD

3.1 预备知识：分布式随机梯度下降算法

分布式 SGD 的执行逻辑如图 2 所示。其中 T

是迭代轮数，η 是学习率，w 是模型参数，X 是训练数据，m 是计算节点个数。

分布式 SGD 是一个迭代的过程，在这个过程中主节点迭代的调度计算节点和服务器节点进行梯度的计算和模型的更新（第 2-4 行）。在计算节点上，计算节点首先从参数服务器拉取最新的模型（第 7 行），然后采样小批量的训练数据计算梯度（第 8，9 行），最后将梯度推送给参数服务器（第 10 行）。在参数服务器上，参数服务器从各个计算节点收取梯度，并且进行模型的更新（第 13，14 行）。

在上述过程中，模型的拉取（第 7 行）和梯度的推送（第 10，13 行）通常会带来比较大的通信开销。因此，Angel 分别从减少通信次数和减少单次通信量这两个方面，提出对应的优化技术来减少通信开销。

3.2 梯度压缩

为了减少单次通信中通信开销，笔者提出了 SketchML[9]，利用数据草图机制来对（键 - 值）表达的梯度进行压缩，从而在不影响模型收敛的情况下，减少每次网络的传输量。

SketchML 算法主要分为三个模块，分别是分位数据草图 - 桶排序量化模块（Quantile-Bucket Quantification）、最小 - 最大数据草图模块（MinMaxSketch）和二进制增量编码模块（Delta-Binary Coding）。前两个模块共同压缩梯度值，第三个模块用来压缩梯度键。SketchML 算法框架主要包括编码和解码两部分。

编码部分：(1) 扫描一遍所有梯度值，建立一个分位数据草图，产生候选的分位值，然后用这些分位值来对梯度值进行桶排序；(2) 每个梯度的值用对应桶内所有梯度值得平均值代替；(3) 用哈希函数计算梯度键的哈希值，把对应桶的索引插入到最小 - 最大数据草图中；(4) 将梯度键转换为增量存储，并且用二进制编码来将增量键压缩为较少的字节。

解码部分：(1) 将增量键恢复为原始的梯度键，然后用恢复的梯度键查询最小 - 最大数据草图，得到梯度值对应的桶索引；(2) 用桶索引来查询得到对应桶的平均值作为梯度值。

3.3 模型平均

另一个减少通信开销的方法是减少通信的频率。基于此想法，Angel+ 提供了基于模型平均 [3] 的解法。模型平均直观的想法是在每次迭代的过程中，我们在局部多次更新模型，然后将多次更新的模型一起推送到参数服务器上，最后由参数服务器对这些模型的更新取平均值，并且更新全局模型。

通过发送模型，而不是发送梯度的方式，Angel+ 允许了计算节点在本地多次更新模型。这样带来的好处是，尽管单次通信的代价没有发生变化，全局通信的次数却可以大大减少。

4 掉队者问题

在真实异构的环境中，常常会存在掉队者问题，即较慢的计算节点会制约整个系统的运行速度。在图 2 算法中，我们发现参数服务器需要集齐所有计算节点的梯度才能更新模型（第 13 行），即当一个计算节点速度很慢时，所有的计算节点和参数服务器节点都只能等待这个计算节点。因此在分布式 SGD 中，掉队者问题是亟需解决的。

4.1 预备知识：同步协议

为了减缓掉队者问题的影响，已有的研究通过各种同步协议放宽了对同步的要求。

严格同步协议（BSP）：在严格同步协议中，每轮迭代之后，所有计算节点都要停止运行，等待其他计算节点完成本轮迭代，才能开始下一轮训练。严格同步协议保证算法的正确性，但是由于计算节点之间需要互相等待，掉队者问题严重。

异步协议（ASP）：异步协议完全不限制计算节

点的速度，每个计算节点完全独立的产生对于模型参数的更新，不用等待其他计算节点就可以开始下一轮，最大化系统的性能。但是异步协议不能保证算法的正确性，有时候模型甚至不会收敛。

有限异步协议（SSP）：有限异步协议是为了权衡严格同步协议和异步协议而提出。有限异步协议规定最快的计算节点和最慢的计算节点的迭代轮数之间的差距不能超过一个阈值。通过限制这个阈值，从而避免出现模型发散的情况，同时缓解掉队者问题带来的影响。

4.2 异构感知的分布式 SGD 算法

尽管 SSP-SGD 和 ASP-SGD 缓解了掉队者带来的系统性能的损失，但是他们将计算节点的局部更新直接累加到全局模型参数上，而没有考虑到掉队者产生的更新具有较大的延迟，而此时的更新对于全局模型参数是异常的，可能会导致不稳定的收敛。

Angel$^+$ 考虑了掉队者带来的陈旧更新的影响，提出了两个异构感知的 SGD 训练算法，分别是 ConSGD 和 DynSGD [10]。ConSGD 借鉴了 BSP 的思想，对每个计算节点的模型更新统一的施加了 1/W 的惩罚项，从而削弱了掉队者带来的影响，这里 W 是计算节点的个数。DynSGD 利用陈旧度（staleness）进一步的考虑了掉队者速度的影响，使得最慢的计算节点的模型更新得到最大的惩罚项，进一步加速了分布式 SGD 的训练过程。

5 机器学习算法实例

为了增强 Angel$^+$ 系统的可用性，我们结合参数服务器和机器学习算法二者的特点，开发了很多高效、易用的机器学习算法，包括逻辑回归、支持向量机、矩阵分解、KMeans、梯度提升树、因子分解机、主题模型、图嵌入、图卷积神经网络等。在本节中，我们以逻辑回归，主题模型以及梯度提升树为例，比较 Angel$^+$ 与其他机器学习系统的性能，参见表 2。

表 2 本文中对比的机器学习任务

Table 2 ML workloads in this paper.

机器学习模型	基线系统	数据集
逻辑回归	Spark, Petuum, TensorFlow	KDD2010
主题模型	LightLDA, Petuum	PubMed
梯度提升树	XGBoost, LightGBM	RCV1

为保证实验的公平性，在每个机器学习模型中，笔者给各个系统设置相同的超参数。实验环境为一个共享的 yarn 集群，网络带宽为 10Gbps.

5.1 逻辑回归

逻辑回归（LR）是一种线性分类模型。由于其简单高效，并且可解释性较强，因此被广泛应用于推荐系统中。

在 Angel$^+$ 分布式逻辑回归的实现中，我们将数据切分到各个计算节点上，而将模型向量切分到多个参数服务器节点上。在进行模型训练的过程中，每个计算节点从各个参数服务器拉取模型，计算梯度并且对梯度进行压缩 [9]，然后将梯度推送回参数服务器。

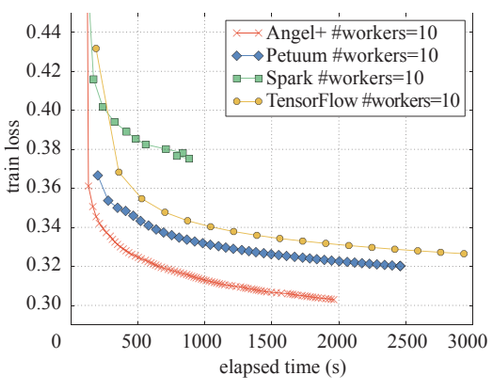


图 3 逻辑回归中，Angel+ 与 Petuum, Spark 以及 TensorFlow 在 KDD2010 数据集上的对比

Fig.3 Angel+ vs. Petuum, Spark, and TensorFlow on logistic regression on KDD2010 dataset

图 3 展示了 Angel$^+$ 与 Spark, Petuum 以及 TensorFlow 在 kdd2010 数据集上的表现。我们可以发现 Angel$^+$ 的性能远远超过了这些基线系统。

5.2 主题模型

主题模型（LDA）是一个应用于语料集合或者离散数据集合的概率生成模型，通常被用来作为文本挖掘。

在分布式 LDA 的实现中，我们将文档 - 话题矩阵 Cd 存储于计算节点上，而将词 - 话题矩阵 Cw 存储于参数服务器上。在已有的分布式实现中，计算节点从参数服务器拉取 Cw，并将 Cw 的更新推送至参数服务器。在 Angel+ 中，我们利用服务器端计算的能力（参见表 1），设计了一个高效分布式 LDA 实现，LDA* [12]。LDA* 综合考虑了各种采样方法的优劣性，提出了一个高效的混合采样器 ;LDA* 也考虑了自然语言长尾的特性，通过将部分长尾词的计算推送到参数服务器上直接进行运算，进而减小计算节点与参数服务器之间的通信开销。

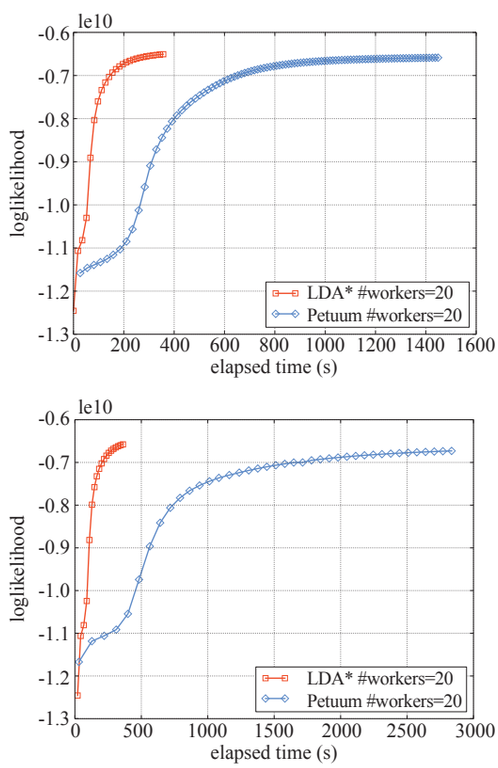


图 4 LDA* 与 LightLDA 以及 Petuum 在 PubMed 数据集上的对比

Fig. 4 LDA* vs. LightLDA and Petuum on PubMed dataset

图 4 展示了 LDA* 和 Petuum 以及 LightLDA[13] 在 PubMed 数据集上的比较，可以发现 LDA* 相比于 LightLDA 和 Petuum 在性能上有较大的提升。

5.3 梯度提升树

梯度提升树（GBDT）[14,15,16] 是一种常用的机器学习模型，该模型不仅能达到高准确率，还具有较好的可解释性。

在分布式 GBDT 的实现中，我们通常需要汇聚不同计算节点上的梯度直方图。随着数据特征维度的增加，同步梯度直方图的通信代价成为了整个系统运行时的瓶颈。为了减少单点通信带来的瓶颈，Angel+ 设计了一个基于参数服务器的分布式梯度提升树算法实现 DimBoost[14]。DimBoost 通过参数服务器实现梯度直方图的同步，从而大大减少了网络的通信代价。DimBoost 还提出了一系列的优化策略，包括两阶段分裂点搜索法，梯度直方图量化压缩，高效地样本 - 树结点索引等，从而提升分布式梯度提升树算法的性能。

图 5 展示了 DimBoost，LightGBM[17]，XGBoost 在 RCV1 数据集上的比较，可以发现 DimBoost 在 RCV1 上能取得最好的性能。

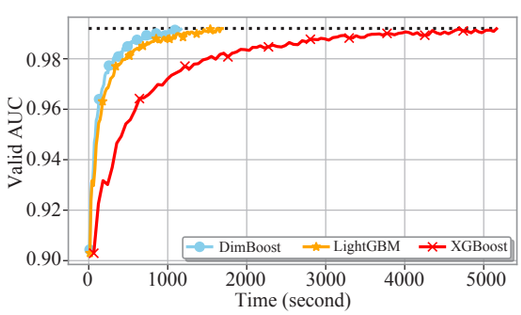


图 5 DimBoost 与 XGBoost, LightGBM 在 RCV1 数据集上的比较

Fig. 5 DimBoost vs. XGBoost, LightGBM, on RCV1 dataset

6 总结与展望

本文介绍了 Angel+，一个基于参数服务器的大

规模机器学习平台。Angel+ 高效的兼容了已有的大数据处理系统比如 Apache Spark 和机器学习系统，比如 PyTorch，使得用户能在统一的平台下高效的使用已有的大数据处理系统和机器学习系统。此外，Angel+ 还针对分布式机器学习系统中通信开销大和掉队者问题分别提出了算法上的优化。为了增强平台的可用性，Angel+ 提供了很多常见机器学习模型的高效实现。

此外，除了机器学习模型的训练，训练数据的特征选择，模型超参数的选择，机器学习模型的选择，模型的集成等也都是机器学习必不可少的部分。因此，我们未来也将在 Angel+ 上层搭建一个 AutoML 系统，方便的帮助用户更好的使用机器学习技术。机器学习平台 Angel+ 的代码已在 Github 开源，也欢迎读者一起使用改进 https://github.com/Angel-ML。

利益冲突声明

所有作者声明不存在利益冲突关系。

参考文献

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, Xiaoqiang Zheng: TensorFlow: A System for Large-Scale Machine Learning [C]. OSDI 2016, 265-283.

[2] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, Zheng Zhang: MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems [C]. CoRRabs/1512.01274.

[3] Eric P. Xing, Qirong Ho, Wei Dai, Jin Kyu Kim, Jinliang Wei, Seunghak Lee, Xun Zheng, Pengtao Xie, Abhimanu Kumar, Yaoliang Yu: Petuum: A New Platform for Distributed Machine Learning on Big Data [C]. KDD 2015, 1335-1344.

[4] Tianqi Chen, Carlos Guestrin: XGBoost: A Scalable Tree Boosting System [C]. KDD 2016, 785-794.

[5] Jie Jiang, Lele Yu, Jiawei Jiang, Yuhong Liu, and Bin Cui: Angel: A new large scale machine learning system [J], NSR 2017, 1-21 .

[6] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J.Shekita, Bor-Yiing Su: Scaling Distributed Machine Learning with the Parameter Server [C]. OSDI 2014, 583-598.

[7] Zhipeng Zhang, Bin Cui, Yingxia Shao, Lele Yu, Jiawei Jiang, Xupeng Miao: PS2: Parameter Server on Spark [C], SIGMOD 2019, 376-388.

[8] Zhipeng Zhang, Jiawei Jiang, Wentao Wu, Ce Zhang, Lele Yu, Bin Cui: MLlib*: Fast Training of GLMs Using Spark MLlib [C], ICDE 2019, 1778-1789.

[9] Jiawei Jiang, Fangcheng Fu, Tong Yang and Bin Cui: SketchML: Accelerating Distributed Machine Learning with Data Sketches [C], SIGMOD 2018, 1269-1284.

[10] Jiawei Jiang, Bin Cui, Ce Zhang and Lele Yu: Heterogeneity-aware Distributed Parameter Servers [C], SIGMOD 2017, 463-478.

[11] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, Ion Stoica: Spark: Cluster Computing with Working Sets [C]. HotCloud 2010.

[12] Lele Yu, Ce Zhang, Yingxia Shao and Bin Cui: LDA*: A Robust and Large-scale Topic Modeling System [C], VLDB 2017, 1406-1417.

[13] Jinhui Yuan, Fei Gao, Qirong Ho, Wei Dai, Jinliang Wei, Xun Zheng, Eric Po Xing, Tie-Yan Liu, Wei-Ying Ma: LightLDA: Big Topic Models on Modest Computer

Clusters[C]. WWW 2015, 1351-1361.

[14] Jiawei Jiang, Bin Cui, Ce Zhang and Fangcheng Fu: DimBoost: Boosting Gradient Boosting Decision Tree to Higher Dimensions [C], SIGMOD 2018, 1363-1376.

[15] Fangcheng Fu, Jiawei Jiang, Yingxia Shao, Bin Cui: An Experimental Evaluation of Large Scale GBDT Systems [C], VLDB 2019.

[16] Jie Jiang, Jiawei Jiang, Bin Cui and Ce Zhang: TencentBoost: A Gradient Boosting Tree System with Parameter Server [C], ICDE 2017, 281-284.

[17] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu: LightGBM: A Highly Efficient Gradient Boosting Decision Tree [C]. NIPS 2017: 3146-3154.

收稿日期：2019年8月15日

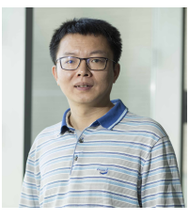
张智鹏，1993年生，北京大学在读博士生。研究方向为分布式机器学习和大数据分析。

本文贡献：系统实现、论文写作。

Zhang Zhipeng, born in 1993, is a PhD candidate from Peking university. His research interests include distributed machine learning and big data analytics.

In this paper he undertakes the following tasks: system implementation and paper writing.

E-mail: zhangzhipeng@pku.edu.cn



崔斌，1975年生，北京大学长江学者特聘教授，计算机系副主任。崔斌博士于2004年在新加坡国立大学获得博士学位。研究方向为数据库系统架构，查询/索引关键技术，大数据分析，分布式机器学习系统等。他是中国计算机学会杰出会员，目前担任数据库专委会秘书长。

本文贡献：系统框架及论文组织。

Cui Bin, born in 1975, is a professor and deputy head of the Department of computer science at Peking University. He obtained his PhD from National University of Singapore in 2004. His research interests include database system architectures, query and index techniques, big data management and distributed machine learning systems.

In this paper he undertakes the following tasks: system framework design and organizing paper structure.

E-mail: bin.cui@pku.edu.cn