УДК 004.852
DOI: 10.15827/0236-235X.138.263-272
Дата подачи статьи: 11.02.22, после доработки: 13.04.22
2022. Т. 35. № 2. С. 263–272

Разработка архитектуры универсального фреймворка федеративного обучения

М.А. Ефремов [1], *аспирант, jakutenshi@gmail.com*
И.И. Холод [1], *д.т.н., доцент, iiholod@mail.ru*

[1] *Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина), г. Санкт-Петербург, 197022, Россия*

Объектом исследования является технология федеративного обучения, которая позволяет осуществлять коллективное машинное обучение на распределенных обучающих наборах данных без их передачи в единое хранилище. Актуальность данной технологии обусловлена, с одной стороны, давно растущим трендом на использование машинного обучения для решения множества прикладных задач, а с другой – ростом запросов, в том числе законодательных, на приватность и обработку данных ближе к источнику или непосредственно на нем.

Основными проблемами при создании систем федеративного обучения являются отсутствие гибких фреймворков для различных сценариев федеративного обучения: большинство существующих решений сосредоточено на обучении искусственных нейронных сетей в централизованной вычислительной среде. Предмет исследования – универсальная архитектура фреймворка для разработки прикладных систем федеративного обучения, позволяющая строить системы для разных сценариев, различных параметров и топологий вычислительной среды, моделей и алгоритмов машинного обучения.

В статье рассмотрена предметная область федеративного обучения, даны основные определения и описан процесс федеративного обучения, приведены и разобраны различные сценарии возможных прикладных задач. Проведен анализ наиболее известных на данный момент фреймворков федеративного обучения, а также их применения для возможных сценариев использования.

В качестве результата описана архитектура универсального фреймворка, который, в отличие от существующих, может быть использован для разработки прикладных систем федеративного обучения разного типа и разных сценариев использования.

Ключевые слова: *машинное обучение, распределенные вычисления, федеративное обучение.*

Сегодня *машинное обучение* (МО) является одной из наиболее важных тем в компьютерных науках и технологиях, в том числе из-за его применения во множестве отраслей. При этом возникает ряд проблем: постоянно растет объем данных, требуется все больше вычислительных мощностей для их анализа, необходимы каналы связи с большей пропускной способностью, растут требования к приватности данных.

В 2017 году компания Google представила концепцию *федеративного обучения* (ФО) [1], сопроводив ее в дальнейшем примером использования подхода для обучения предсказательной модели набора текстов на клавиатуре GBoard для Android [2]. ФО является разновидностью МО на распределенных данных, в ходе которого между участниками передаются только параметры моделей МО, а сами данные остаются на устройствах владельцев.

Все множество вычислительных узлов системы, участвующих в процессе ФО, называется федерацией.

Процесс ФО начинается с инициализации глобальной модели на сервере. Затем она рассылается на клиенты, где происходит обучение модели на локальных данных. После этого все модели собираются на сервере и выполняется их агрегация в единую глобальную модель. Для нейронных сетей широко используются алгоритмы агрегации FedSGD [1], FedAvg [1], FedMA [3], а для деревьев решений – Federboost [4].

В данной работе рассматривается реализация ФО в виде фреймворка для разработки прикладных систем обучения разного типа. Предметом исследования является архитектура фреймворка для разработки прикладных систем ФО. Основная проблема при разработке систем ФО заключается в отсутствии универсальных фреймворков, которые могут использоваться для реализации систем ФО разного типа. Наиболее важные характеристики типов систем ФО – топология и масштабирование федерации, а также способ разделения данных между клиентами. Результатом является архи-

тектура фреймворка ФО, предоставляющая возможность гибкой реализации различных сценариев обучения.

Обзор существующих решений

На рисунке 1 показана таксономия систем ФО, в соответствии с которой далее будут рассмотрены некоторые существующие фреймворки [5, 6]. Таксоны масштабирования федераций и способа размещения данных – наиболее важные элементы при выборе сценариев и инструментов реализации систем ФО.

Обычно в ФО используются три типа топологии:

– централизованная, когда существует один сервер агрегации данных, а клиенты федерации подключены напрямую к нему;

– иерархическая, когда клиенты федерации подключены к серверу через промежуточные узлы, которые могут выполнять функцию как клиента, так и сервера;

– децентрализованная, когда каждый узел выполняет функции и клиента, и сервера.

Последний иногда относят к отдельной категории – роевое обучение [7].

Принято выделять два типа масштабирования федераций:

– кроссхранилищное, подразумевающее использование относительно небольшого количества (2–100) практически всегда доступных хранилищ данных в качестве клиентов;

– кроссустройственное [8], подразумевающее большое количество (более тысячи) нестабильно присутствующих в федерации устройств с данными.

В ФО принято выделять два способа размещения данных:

– горизонтальный, когда на разных клиентах ФО хранятся разные подмножества векторов исходного набора данных с одинаковым множеством атрибутов;

– вертикальный, когда на разных клиентах ФО хранится одинаковое множество векторов исходного набора данных с разными подмножествами атрибутов.

Для разных прикладных сценариев использования ФО требуется реализация разных типов систем. В таблице 1 приведены основные сценарии, соответствующие описанным выше характеристикам. Таким образом, возможность реализации на основе фреймворка ФО того или иного типа системы означает возможность решения соответствующей прикладной задачи.

Рис. 1. Таксономия федеративного обучения

Fig. 1. Taxonomy of federated learning

Существует несколько разработанных фреймворков для ФО, некоторые из них были подробно рассмотрены в [9]. Общей характеристикой всех решений является ориентированность на Python в качестве языка и платформы разработки, без открытых инструментов обучения моделей на смартфонах.

• TensorFlow Federated (TFF) от Google на данный момент поддерживает работу только в режиме симуляции с кроссхранилищным разделением данных. В качестве моделей предлагаются нейронные сети, реализуемые фреймворком TensorFlow (TF) [10]. Несмотря на то, что TFF от Google представляют как пример работы ФО приложение для Android – клавиатуру GBoard, исходный код программы закрыт, а в открытом доступе нет ни реального режима фреймворка, ни библиотек для непосредственного обучения хотя бы моделей нейронных сетей.

• PySyft от сообщества OpenMind в отличие от TFF поддерживает реальный режим работы, а также поддерживает основные методы дополнительного повышения приватности вычислений при горизонтальном разделении данных. Тем не менее, в качестве моделей он также использует только нейронные сети с использованием библиотек TensorFlow и PyTorch (PT) [11].

• FATE разрабатывается китайским онлайн-банком WeBank от Tencent. Как и PySyft, поддерживает при горизонтальном распределении все основные методы повышения приватности, но позволяет использовать и модели МО, отличные от нейронных сетей, например, деревья решений с алгоритмом градиентного бустинга. FATE поддерживает вертикальное разделение данных, но применяются алгоритмы, отличные от используемых для горизонтального распределения для одних и тех же моделей МО. В качестве библиотек для нейронных

Таблица 1

Возможные сценарии применения ФО

Table 1

Possible federated learning usage cases

Размещение	Топология			
	Централизованная/Иерархическая		Децентрализованная	
	Кроссхранилищное масштабирование	**Кроссустройственное масштабирование**	**Кроссхранилищное масштабирование**	**Кроссустройственное масштабирование**
Горизонтальное	Сценарий больших компаний, предоставляющих МО через облачные сервисы, но заинтересовавшихся ФО из-за законов о приватности данных	Сценарий систем интернета вещей, граничных вычислений, когда используются небольшие промежуточные устройства накопления данных в реальном времени	Сценарий для независимых организаций с собственными хранилищами и серверами, но желающих разрабатывать модели МО, используя данные всех участников, не раскрывая сами данные	Сценарий распределенных пользовательских персональных данных, например, из интернет-браузеров, позволяющих извлекать маркетинговую информацию, не раскрывая данные пользователя
Вертикальное	Сценарий кооперации нескольких больших компаний, например, из медицинской и банковской сфер для оформления кредита без обязательного страхования жизни на основе общей модели о финансовом и медицинском состояниях заемщика	Сценарий использования данных с устройств-датчиков, получающих информацию об одном объекте, но с разных точек зрения, с помощью различных метрик	Сценарий похож на аналогичный вертикальный иерархический	Сценарий, аналогичный примеру с браузером, но позволяющий совмещать текущий подход к интернет-рекламе с уникальным идентификатором и децентрализованное ФО без передачи данных из браузера пользователя

сетей могут использоваться TensorFlow и PyTorch [12].

• PaddleFL тоже разрабатывается китайскими исследователями из банковской сферы от Baidu. Основным отличием фреймворка является использование очередей сообщений на основе ZeroMQ вместо удаленного вызова методов при помощи gRPC, как это и реализовано в большинстве разработок. Также стоит отметить использование собственной библиотеки глубокого обучения PaddlePaddle (PP). Заявлены и поддерживаются вертикальное распределение данных и децентрализованная топология на основе протокола ABY3 (Arithmetic, Boolean and Yao secret shares) [13], но с использованием только трех узлов.

• FEDn разрабатывается шведской компанией Scaleout Systems AB. Ее основным отличием является независимость клиентов от платформы, языка разработки, фреймворка МО или технологии передачи данных. Предоставлены реализации реального режима работы при помощи gRPC и обучения только нейронных сетей с библиотеками TensorFlow и PyTorch. Также стоит отметить выделение отдельного вычислительного элемента для агрегации моделей с присоединенных федеративных клиентов. Таким образом, FEDn позволяет реализовывать как централизованную, так и иерархическую топологию федерации [14].

• Flower разрабатывается немецкой компанией Adap GmbH. Фреймворк во многом похож на FEDn по архитектурным решениям, но поддерживает только централизованное ФО. Несмотря на это, разработчики отдельно выделяют реализацию клиентов под разные платформы, такие как Android-смартфоны, или граничные устройства, такие как Nvidia Jetson [15].

Сравнение описанных фреймворков отражено в таблице 2. Можно заметить, что наибольшее покрытие различных типов систем ФО реализовано в PaddleFL, однако даже он не может быть использован для разработки всех типов систем ФО.

В данной статье рассматривается архитектура фреймворка Federated Learning for Java (FL4J) [16] для разработки прикладных систем

Таблица 2

Сравнение фреймворков ФО

Table 2

Comparison of federated learning frameworks

Таксоны	Тип системы ФО						
	TFF	PySyft	FATE	FEDn	Flower	PaddleFL	FL4J
Масштабирование							
Кроссхранилищное	+	+	+	+	+	+	+
Кроссустройственное	-	+	-	+	+	-	+
Топология							
Централизованная	+	+	+	+	+	+	+
Иерархическая	-	-	-	+	-	-	+
Децентрализованная	-	-	-	-	-	~	+
Размещение							
Горизонтальное	+	+	+	+	+	+	+
Вертикальное	-	-	+	-	-	~	+

ФО разного типа. Одним из отличий FL4J является сфокусированность на платформе Java Virtual Machine (JVM). В качестве реализации нейронных сетей используется библиотека DeepLearning4Java (DL4J) [17, 18]. Данное решение позволило практически без изменения исходного кода перенести реализацию клиента ФО на смартфон под управлением Android. Таким образом, решение об использовании JVM как основной платформы, а также использование фреймворка DL4J позволили проводить исследования с участием смартфонов под управлением операционной системы Android. Выбор программной платформы – не единственное важное отличие FL4J от альтернатив. Опишем, за счет каких архитектурных решений FL4J достигается универсальность в реализации на его основе систем ФО различного типа.

Архитектура решения

Основные модули фреймворка FL4J. Фреймворк FL4J создавался с целью предоставить набор абстракций и интерфейсов, достаточно общих, чтобы реализовывать системы ФО под самые разные задачи пользователей. Можно выделить две основные группы пользователей фреймворка:

– разработчики систем обработки больших данных, подходящие к проблеме как к распределенной системе;

– разработчики алгоритмов МО, исследующие алгоритмы и модели МО.

На рисунке 2 представлена схема с ролями и примером реализации пакетов фреймворка, которые могут заинтересовать разработчиков систем обработки больших данных и исследователей алгоритмов МО.

Для разработчиков систем обработки больших данных наибольший интерес представляют сущности, отвечающие за представление вычислительных узлов (FLService, FLServer и FLClient), а также FLRouter – общий интерфейс для реализаций межпроцессного взаимодействия между узлами. Для исследователей МО наиболее интересны пакеты для моделей МО, алгоритмов и настроек: model, algorithm и settings соответственно. Общим пакетом для обеих категорий пользователей является data, в котором реализованы сущности для взаимодействия с источниками наборов данных.

Практически все приведенные выше пакеты содержат классы, которые являются интерфейсами или абстрактными классами, доступными для расширения. Одна из целей при проектировании FL4J – предоставить возможности для ис-

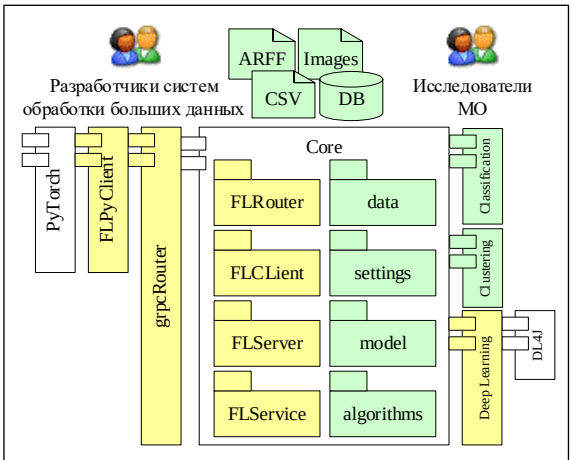

Рис. 2. Пользователи фреймворка и основные модули согласно ролям

Fig. 2. Framework users and common modules according to the roles

пользования и развития не только для существующих алгоритмов ФО, но и для новых. Таким образом, обе категории пользователей фреймворка имеют должную гибкость. На рисунке 3 представлена диаграмма, показывающая отношения между базовыми сущностями фреймворка.

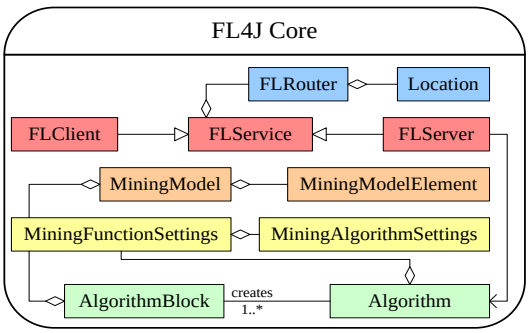

Рис. 3. Отношения между базовыми сущностями FL4J

Fig. 3. Relations between base entities from FL4J

Вычислительными элементами в федерации выступают экземпляры класса FLService. Существуют две реализации с разными назначениями: FLServer и FLClient. Главная задача сервера – принять запрос на ФО от пользователя системы, спланировать и отправить задачу обучения на исполнение, а по готовности вернуть обученную модель. Задача клиентов – исполнять задачи ФО, полученные от сервера. Разница в целях отражается и в структуре классов: сервер содержит фабрику планировщиков, а клиент – хэш-таблицу с физическим представлением наборов данных по имени набора.

Блок алгоритма представляет собой интерфейс для любого объекта-преобразователя модели. Модель может быть преобразована с данными и без входящих данных. Фреймворк предоставляет несколько базовых блоков, соответствующих привычным в программировании элементам, таким как блок ветвления или блок цикла с условием по векторам данных. Разработчики алгоритмов могут реализовать собственные блоки алгоритма с требуемой детализацией и разнообразием. Кроме того, разбиение алгоритма на множество отдельных блоков позволяет использовать ряд операций из области оптимизаций компиляции, таких как перестановка и расщепление циклов (loop interchange и loop fission), для преобразования разработанного алгоритма между горизонтальным и вертикальным разделением данных [19].

Все блоки алгоритма объединяются в блок последовательности – SequenceBlock. Такой блок имеет два дополнительных флага: могут ли вложенные блоки выполняться параллельно и должно ли исполнение производиться в разделяемой памяти. Данные флаги требуются для планировщика исполнения алгоритма. Планирование исполнения – процесс сопоставления каждого блока алгоритма с узлами федерации, а также генерации правил преобразования для исполнения блока на отобранных узлах. Каждый кортеж такого сопоставления, состоящий из блока, узла и множества правил, называется подпланом. Итоговый план исполнения является иерархией подпланов. Иерархичность плана возникает из-за возможности параллельно выполнять последовательность блоков алгоритма на множестве узлов, подходящих под заданные условия.

Каждая из сущностей так или иначе отвечает за реализацию описанных ранее характеристик ФО. Рассмотрим, как реализованные в библиотеке архитектурные решения позволяют построить различные системы ФО в соответствии с приведенной выше таксономией.

Реализация для горизонтально и вертикально разделенных данных

Разделение данных необходимо учитывать на разных этапах обучения:

– при чтении физических данных, а также их логическом представлении и преобразовании для МО;

– при выполнении алгоритмов МО, которые должны обрабатывать только часть горизонтально или вертикально разделенных данных;

– при агрегации моделей МО, обученных на разделенных наборах.

Для горизонтального разделения данных требуется выбрать нужные узлы и составить для них множество преобразований. Алгоритмы обучения при этом не требуют модификации: модель обновляется на всех найденных узлах, затем агрегируется в одну, глобальную. Вертикальный сценарий требует такого изменения алгоритма обучения, чтобы модель могла частично обновляться на узлах в зависимости от доступных атрибутов на каждом конкретном узле. Таким образом, требуется механизм описания модифицированных алгоритмов, а также планирования исполнения обучения. С этой целью FL4J использует представление алгоритмов в виде иерархий блоков алгоритма для каждого из этапов или сценариев ФО: для инициализации и оценки модели

как общих этапов обучения, а также для сценариев симуляционного, горизонтального и вертикального исполнения алгоритма обучения.

Для планирования исполнения алгоритмов могут подбираться различные стратегии в зависимости от поставленных целей: равномерная загруженность федерации, наиболее полное использование данных, максимальная пропускная способность и др. Таким образом, используя комбинацию из интерфейсов для описания и преобразования данных, описания алгоритмов для разных сценариев, а также планирования исполнения с различными стратегиями, все эти архитектурные решения позволяют реализовать самые разные задачи и сценарии ФО как для горизонтального, так и для вертикального разделения данных.

Реализация разных топологий вычислительных узлов

Для реализации разных топологий в FL4J базовым классом служит интерфейс FLRouter, определяющий три основных метода взаимодействия между узлами:

– получение описания конкретного узла по идентификатору;

– получение информации о всей федерации;

– выполнение подплана.

Структурно именно в FLRouter хранятся список доверенных узлов для выполнения запросов и список узлов, для которых можно создавать запросы. Так реализуется графовая модель связей между узлами. С одной стороны, данный подход имеет ряд проблем: необходимость проверки на внутренние циклы или механизмы предотвращения появления блуждающих запросов, возникновение несвязности федерации в случае потери связующих узлов и т.д. С другой стороны, наиболее распространенными топологиями является либо один сервер с множеством напрямую подключенных клиентов, либо дерево с корнями-серверами и узлами-клиентами. На рисунках 4–6 изображены примеры топологий, демонстрирующие, что любую из них можно представить в виде графа, а наиболее используемые подходы по своей структуре подразумевают отсутствие многих проблем графов из общего случая. Тем не менее, поскольку FLService – интерфейс, допустимы самые различные реализации алгоритмов взаимодействия, в том числе учитывающие общую графовую структуру федерации, опираясь на уже известные подходы при маршрутизации пакетов в компьютерной сети, такие как параметр TTL у запроса.

Реализация разных типов масштабирования

Аналогично планировщику одна из целей при проектировании FLRouter – простота расширения под различные сценарии и технологии. Разные сценарии имеют разные критерии оценки производительности. Два наиболее характерных сценария – кроссхранилищное и кроссустройственное исполнение ФО. Для кроссхранилищного случая не требуются затратные алгоритмы исполнения запросов между узлами: хранилища данных соединены напрямую с сервером и клиентом агрегации. При этом важным критерием кроссхранилищной системы будет скорость обработки запросов в секунду. Для данного случая создана реализация GrpcRouter, использующая бинарный формат сообщений protobuf и технологию

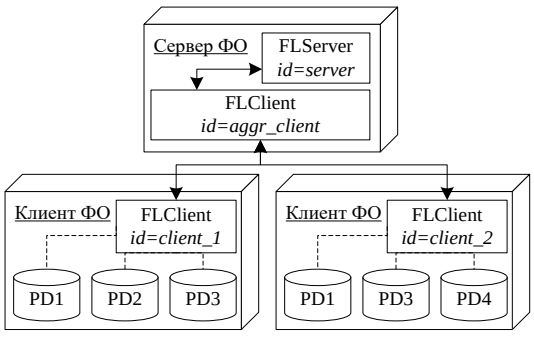

Рис. 4. Централизованная топология узлов

Fig. 4. A centralized topology of the services

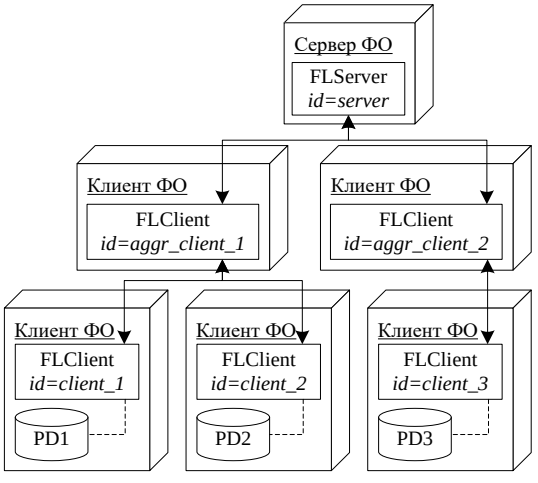

Рис. 5. Иерархическая топология узлов

Fig. 5. A hierarchical topology of the services

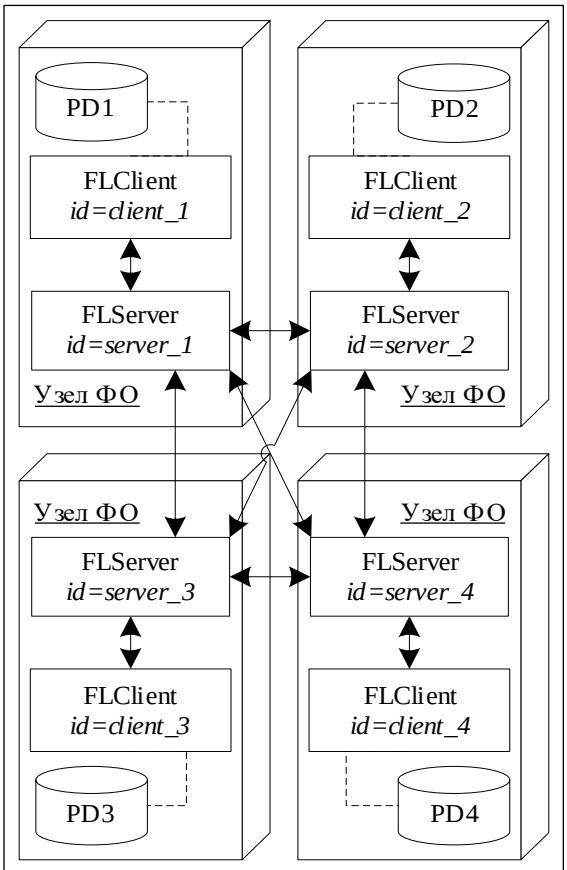

Рис. 6. Децентрализованная топология узлов

Fig. 6. A decentralized topology of the services

gRPC, а также неблокирующие и асинхронные алгоритмы для параллельной многопоточной обработки запросов.

Другим сценарием является кроссустройственное исполнение. Для такого случая характерны нестабильность и большое количество маломощных устройств, на которых размещены федеративные узлы-клиенты. В таком случае основным критерием является устойчивость системы к ошибкам и отказу узлов системы. Скорость в данном случае не является критическим параметром: приводившийся ранее пример с клавиатурой GBoard [2] задействует устройство в обучении при подключении смартфона к зарядному устройству. Таким образом, многие устройства могут не участвовать в вычислении, даже если планировщик добавил узел в план исполнения алгоритма. Кроме того, возможны дополнительные технические особенности для различных устройств. Например, стандартный сервер gRPC не поддерживается для запуска в приложениях Android. Тем не менее, для обеспечения взаимодействия достаточно было реализовать три основных метода взаимодействия на основе встроенного HTTP-сервера с JSON-форматом сообщений. RestRouter реализован на основе очередей исполнения с ограниченным размером, асинхронного взаимодействия между узлами, а также с фокусом на обработке различных ответов HTTP. Такой подход позволяет работать с нестабильными маломощными устройствами, выполняя каждый запрос по доступности узла или поддерева.

Наилучшие результаты для данных сценариев ФО будут достигаться при правильном подборе реализации планировщика, учитывающего вероятность доступности узлов при составлении плана, а также наиболее подходящих реализаций межпроцессного взаимодействия на клиентских устройствах. Архитектура FL4J позволяет не просто реализовывать кроссхранилищные и кроссустройственные сценарии, но и управлять характеристиками системы ФО как частного случая распределенных систем.

Заключение

В результате исследования была разработана архитектура универсального фреймворка ФО FL4J, покрывающего все типы систем ФО. Универсальность фреймворка обеспечивается за счет следующих решений:

– гибкость масштабирования и топологии достигается за счет графовой модели системы вычислительных элементов, а также механизмов планирования исполнения обучения;

– гибкость разработки моделей МО достигается использованием абстрактных моделей со встроенными методами агрегации по проиндексированным элементам модели;

– гибкость реализации обучения на горизонтально и вертикально размещенных данных достигается за счет физического и логического представления наборов данных, наличия сущностей для описания произвольных алгоритмов, а также планирования исполнения обучения;

– гибкость реализации различных режимов работы систем ФО и технологий передачи данных достигается за счет выделения отдельного интерфейса для реализации протокола передачи данных.

Из недостатков можно отметить сложность алгоритмов для реализации общих и абстрактных решений. В то же время для наибольшей производительности для каждого множества

сценариев требуется определенное подмножество наиболее подходящих реализаций, допускающих ряд упрощений, исходя из ограничений решаемых задач.

Фреймворк активно развивается: добавляются новые алгоритмы, модели, реализации межузловых взаимодействий, а также клиентов для других платформ, например Python, из-за обилия таких библиотек для глубокого обучения, как TensorFlow и PyTorch. С целью дальнейшего развития фреймворка планируются исследования обучения с подкреплением.

Литература

1. McMahan H.B., Moore E., Ramage D., Hampson S., Arcas B.A.Y. Communication-efficient learning of deep networks from decentralized data. AISTATS, 2017, pp. 1273–1282.

2. Hard A., Rao K., Mathews R., Ramaswamy S., Beaufays F. et al. Federated learning for mobile keyboard prediction. ArXiv, 2018, art. 1811.03604. URL: https://arxiv.org/pdf/1811.03604.pdf (дата обращения: 15.01.2022).

3. Wang H., Yurochkin M., Sun Y., Papailiopoulos D., Khazaeni Y. Federated learning with matched averaging. ArXiv, 2020, art. 2002.06440. URL: https://arxiv.org/abs/2002.06440 (дата обращения: 15.01.2022).

4. Tian Z., Zhang R., Hou X., Liu J., Ren K. Federboost: Private federated learning for GBDT. ArXiv, 2020, art. 2011.02796. URL: https://arxiv.org/pdf/2011.02796.pdf (дата обращения: 15.01.2022).

5. Yin X., Zhu Y., Hu J. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. ACM CSUR, 2021, vol. 54, no. 6, pp. 1–36. DOI 10.1145/3460427.

6. Khan L.U., Saad W., Han Z., Hossain E., Hong C.S. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. IEEE Communications Surveys & Tutorials, 2021, vol. 23, no. 3, pp. 1759–1799. DOI: 10.1109/COMST.2021.3090430.

7. Warnat-Herresthal S., Schultze H., Shastry K.L., Manamohan S. et al. Swarm learning for decentralized and confidential clinical machine learning. Nature, 2021, vol. 594, no. 7862, pp. 265–270. DOI: 10.1038/s41586-021-03583-3.

8. Kairouz P., McMahan H.B., Avent B., Bellet A. et al. Advances and open problems in federated learning. ArXiv, 2019, art. 1912.04977. URL: https://arxiv.org/pdf/1912.04977.pdf (дата обращения: 15.01.2022).

9. Kholod I.I., Yanaki E., Fomichev D., Shalugin E. et al. Open-source federated learning frameworks for IoT: A comparative review and analysis. Sensors, 2021, vol. 21, no. 1, art. 167. DOI: 10.3390/s21010167.

10. Bonawitz K., Yanaki E., Fomichev D., Shalugin E., Novikova E., Filippov E., Nordlund M. Towards federated learning at scale: System design. Proc. II Conf. of Machine Learning and Systems, 2019, vol. 1, pp. 374–388.

11. Ziller A., Trask A., Lopardo A., Szymkow B. et al. PySyft: A library for easy federated learning. In: Federated Learning Systems, 2021, pp. 111–139. DOI: 10.1007/978-3-030-70604-3_5.

12. Liu Y., Trask A., Lopardo A., Szymkow B. et al. FATE: An industrial grade platform for collaborative learning with data protection. J. of Machine Learning Research, 2021, vol. 22, no. 226, pp. 1–6.

13. Ma Y., Yu D., Wu T., Wang H. PaddlePaddle: An open-source deep learning platform from industrial practice. Frontiers of Data and Domputing, 2019, vol. 1, no. 1, pp. 105–115. DOI: 10.11871/jfdc.issn.2096.742X.2019.01.011.

14. Ekmefjord M., Ait-Mlouk A., Alawadi S., Åkesson M. et al. Scalable federated machine learning with FEDn. ArXiv, 2021, art. 2103.00148. URL: https://arxiv.org/pdf/2103.00148.pdf (дата обращения: 15.01.2022).

15. Beutel D.J., Topal T., Mathur A., Qiu X., Titouan P., de Gusmao P.P.B., Lane N.D. Flower: A friendly federated learning research framework. ArXiv, 2020, art. 2007.14390. URL: https://akhilmathurs.github.io/papers/beutel_flower2020.pdf (дата обращения: 15.01.2022).

16. Efremov M.A., Kholod I.I., Kolpaschikov M.A. Java federated learning framework architecture. Proc. IEEE ElConRus, 2021, pp. 306–309. DOI: 10.1109/ElConRus51938.2021.9396508.

17. Nguyen G., Dlugolinsky S., Bobák M., Tran V. et al. Machine learning and deep learning frameworks and libraries for large-scale data mining: A survey. Artificial Intelligence Review, 2019, vol. 52, no. 1, pp. 77–124. DOI: 10.1007/s10462-018-09679-z.

18. Parvat A., Chavan J., Kadam S., Dev S., Pathak V. A survey of deep-learning frameworks. Proc. XVII ICISC, 2017, pp. 1–7. DOI: 10.1109/ICISC.2017.8068684.

19. Kholod I., Shorov A., Gorlatch S. Efficient distribution and processing of data for parallelizing data mining in mobile clouds. JoWUA, 2020, vol. 11, no. 1, pp. 2–17. DOI: 10.22667/JOWUA.2020.03.31.002.

Software & Systems
DOI: 10.15827/0236-235X.138.263-272

Received 11.02.22, Revised 13.04.22
2022, vol. 35, no. 2, pp. 263–272

Developing universal framework design for federated learning

M.A. Efremov [1], *Postgraduate Student, jakutenshi@gmail.com*
I.I. Kholod [1], *Dr.Sc. (Engineering), Associate Professor, iiholod@mail.ru*

[1] *St. Petersburg Electrotechnical University "LETI", St. Petersburg, 197022, Russian Federation*

Abstract. The paper researches the technology of federated learning that allows collective machine learning on distributed training datasets without transferring them to a single central storage. The relevance of the technology is determined by the long growing trend towards using machine learning methods to solve many applied problems on the one hand, and by the growth of requests for privacy and data processing closer to the data source or directly at the source, including legislative ones, on the other hand.

The main problems in creating federated learning systems are the lack of flexible frameworks for various federated learning scenarios: the majority of the existing solutions focus on training artificial neural networks in a centralized computing environment. The subject of the research is the common framework architecture for developing applied federated learning systems, which allows building systems for different scenarios, parameters and topologies of the computing environment, various models, and machine learning algorithms.

The article considers the federated learning subject area, gives the main definitions, describes the process of federated learning, presents and analyzes various scenarios of possible applied tasks for federated learning. It contains the analysis of the most well-known federated learning frameworks at the time of writing, as well as their application for possible cases that were described previously.

As a result, there is a description of the architecture of a universal framework that, unlike the existing ones, can be used to develop applied federated learning systems of various types and different cases.

Keywords: machine learning, distributed computing, federated learning.

References

1. McMahan H.B., Moore E., Ramage D., Hampson S., Arcas B.A.Y. Communication-efficient learning of deep networks from decentralized data. *AISTATS*, 2017, pp. 1273–1282.
2. Hard A., Rao K., Mathews R., Ramaswamy S., Beaufays F. et al. Federated learning for mobile keyboard prediction. *ArXiv*, 2018, art. 1811.03604. Available at: https://arxiv.org/pdf/1811.03604.pdf (accessed January 15, 2022).
3. Wang H., Yurochkin M., Sun Y., Papailiopoulos D., Khazaeni Y. Federated learning with matched averaging. *ArXiv*, 2020, art. 2002.06440. Available at: https://arxiv.org/abs/2002.06440 (accessed January 15, 2022).
4. Tian Z., Zhang R., Hou X., Liu J., Ren K. Federboost: Private federated learning for GBDT. *ArXiv*, 2020, art. 2011.02796. Available at: https://arxiv.org/pdf/2011.02796.pdf (accessed January 15, 2022).
5. Yin X., Zhu Y., Hu J. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM CSUR*, 2021, vol. 54, no. 6, pp. 1–36. DOI 10.1145/3460427.
6. Khan L.U., Saad W., Han Z., Hossain E., Hong C.S. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Communications Surveys & Tutorials*, 2021, vol. 23, no. 3, pp. 1759–1799. DOI: 10.1109/COMST.2021.3090430.
7. Warnat-Herresthal S., Schultze H., Shastry K.L., Manamohan S. et al. Swarm learning for decentralized and confidential clinical machine learning. *Nature*, 2021, vol. 594, no. 7862, pp. 265–270. DOI: 10.1038/s41586-021-03583-3.
8. Kairouz P., McMahan H.B., Avent B., Bellet A. et al. Advances and open problems in federated learning. *ArXiv*, 2019, art. 1912.04977. Available at: https://arxiv.org/pdf/1912.04977.pdf (accessed January 15, 2022).
9. Kholod I.I., Yanaki E., Fomichev D., Shalugin E. et al. Open-source federated learning frameworks for IoT: A comparative review and analysis. *Sensors*, 2021, vol. 21, no. 1, art. 167. DOI: 10.3390/s21010167.
10. Bonawitz K., Yanaki E., Fomichev D., Shalugin E., Novikova E., Filippov E., Nordlund M. Towards federated learning at scale: System design. *Proc. II Conf. of Machine Learning and Systems*, 2019, vol. 1, pp. 374–388.
11. Ziller A., Trask A., Lopardo A., Szymkow B. et al. PySyft: A library for easy federated learning. In: *Federated Learning Systems*, 2021, pp. 111–139. DOI: 10.1007/978-3-030-70604-3_5.

12. Liu Y., Trask A., Lopardo A., Szymkow B. et al. FATE: An industrial grade platform for collaborative learning with data protection. *J. of Machine Learning Research*, 2021, vol. 22, no. 226, pp. 1–6.

13. Ma Y., Yu D., Wu T., Wang H. PaddlePaddle: An open-source deep learning platform from industrial practice. *Frontiers of Data and Domputing*, 2019, vol. 1, no. 1, pp. 105–115. DOI: 10.11871/jfdc.issn.2096.742X.2019.01.011.

14. Ekmefjord M., Ait-Mlouk A., Alawadi S., Åkesson M. et al. Scalable federated machine learning with FEDn. *ArXiv*, 2021, art. 2103.00148. Available at: https://arxiv.org/pdf/2103.00148.pdf (accessed January 15, 2022).

15. Beutel D.J., Topal T., Mathur A., Qiu X., Titouan P., de Gusmao P.P.B., Lane N.D. Flower: A friendly federated learning research framework. *ArXiv*, 2020, art. 2007.14390. Available at: https://akhilmathurs.github.io/papers/beutel_flower2020.pdf (accessed January 15, 2022).

16. Efremov M.A., Kholod I.I., Kolpaschikov M.A. Java federated learning framework architecture. *Proc. IEEE ElConRus*, 2021, pp. 306–309. DOI: 10.1109/ElConRus51938.2021.9396508.

17. Nguyen G., Dlugolinsky S., Bobák M., Tran V. et al. Machine learning and deep learning frameworks and libraries for large-scale data mining: A survey. *Artificial Intelligence Review*, 2019, vol. 52, no. 1, pp. 77–124. DOI: 10.1007/s10462-018-09679-z.

18. Parvat A., Chavan J., Kadam S., Dev S., Pathak V. A survey of deep-learning frameworks. *Proc. XVII ICISC*, 2017, pp. 1–7. DOI: 10.1109/ICISC.2017.8068684.

19. Kholod I., Shorov A., Gorlatch S. Efficient distribution and processing of data for parallelizing data mining in mobile clouds. *JoWUA*, 2020, vol. 11, no. 1, pp. 2–17. DOI: 10.22667/JOWUA.2020.03.31.002.

Для цитирования

Ефремов М.А., Холод И.И. Разработка архитектуры универсального фреймворка федеративного обучения // Программные продукты и системы. 2022. Т. 35. № 2. С. 263–272. DOI: 10.15827/0236-235X.138.263-272.

For citation

Efremov M.A., Kholod I.I. Developing universal framework design for federated learning. *Software & Systems*, 2022, vol. 35, no. 2, pp. 263–272 (in Russ.). DOI: 10.15827/0236-235X.138.263-272.