Новоженов Владимир Алексеевич, магистр

Ковтушенко Александр Петрович, к.ф-м.н., доцент

*МГТУ им. Н. Э. Баумана, кафедра* *«Программное обеспечение ЭВМ и информационные технологии». Москва, Российская Федерация*

**Среда обработки заданий, посылаемых на кластер из внешнего приложения для ускорения вычислений**

**Аннотация**

В представленной статье проводится анализ и формирование целевого взаимодействия системы обработки заданий на кластере и самого кластера, применяемого для ускорения вычислений в решении класса задач, связанного с обучением нейронных сетей.

**Ключевые слова:** кластерные вычисления, нейронные сети, распределенные вычисления, глубокое обучение, среда обработки, ускорение вычислений.

**Abstract**

In this article, we analyze and formulate the target interaction of the job processing system on the cluster and the cluster itself, which is used to speed up computations in solving the class of problems associated with training of neural networks.

**Keywords:** cluster computing, neural networks, distributed computing, deep learning, processing environment, acceleration of computing.

**Введение**

В настоящее время наблюдается ускоренный рост приложений и программных решений, использующих алгоритмы машинного и глубокого обучения. Перед использованием моделей машинного и глубокого обучения их необходимо «натренировать» на большом объеме данных. Эта «тренировка» или «обучение» представляет из себя огромное количество математических операций и преобразований, в результате которых алгоритмы получают веса, с помощью которых могут впоследствии делать прогнозы, распознавать объекты на фотографиях и видеопотоке, переводить разговорную речь в печатный текст с высокой точностью. Для обучения таких алгоритмов требуются большие объемы данных (зачастую измеряющиеся в Гигабайтах или даже Терабайтах). Вследствие большого количества математических операций для произведения обучения даже на одном объекте, а также огромного количества необходимых данных (объектов) – «обучение» может происходить в течение продолжительного времени - на мощнейших CPU это может занимать недели и даже месяцы. С использованием GPU время обучения может значительно сокращаться благодаря большему количеству блоков арифметико-логических устройств, которые эффективно и быстро производят вычисления матричных операций. Однако даже с использованием нескольких GPU ускорителей на одной вычислительной машине сложно достичь хорошей скорости вычислений.

В настоящее время эта проблема решается специализированными и промышленными GPU-ускорителями, а также с помощью кластерных вычислений [1]. Многие современные фреймворки для машинного и глубокого обучения также имеют встроенную поддержку распределенных кластерных вычислений (pytorch.distributed) или же существуют программные библиотеки, используя которые возможно обучать нейронные сети и классические алгоритмы машинного обучения (такие как градиентный бустинг) распределенно (horovod, dask).

Единожды обученные нейронные сети имеют свойство терять свои качества со временем из-за так называемого “concept drift” – когда входные условия, на которых обучалась нейронная сеть со временем изменяются (например, модель, распознающая лица со временем, станет хуже работать – так как лица людей также меняются со временем. Или же модель, прогнозирующая котировки акций, станет хуже прогнозировать с появлением новых игроков на рынке. Или же NLP-модель станет хуже работать, так как со временем в языке появляются новые слова, выражения, меняется сленг и т.д.).[2]

Поэтому их периодически необходимо «дообучать» или же обучать заново. Что также может занять большое количество времени, причем настолько большое, что к концу обучения, модель снова станет неактуальной и будет нуждаться в повторном обучении вновь. Поэтому столь актуально применение значительных вычислительных мощностей, коими могут являться вычислительные кластеры.

Однако построение процесса дообучения моделей (как вычислительных задач, решаемых на кластере с использованием некоторого внешнего программного интерфейса) может также стать нетривиальной задачей.

**Проектирование и построение системы обработки заданий**

Рассмотрим весь жизненный цикл действующей модели глубокого обучения на примере следующей системы: на рабочих компьютерах пользователей установлено программное обеспечение, обрабатывающее нажатия клавиш клавиатуры, считывающее продолжительность нажатий на клавиши, время между нажатиями разных клавиш, порядок нажатий на клавиши – то есть «клавиатурный почерк» пользователя. Из собранной информации формируется сообщение и отправляется на вход нейронной сети, которая делает вывод, является ли пользователь тем, за кого себя выдает. В случае если данная проверка заканчивается неудачей – блокирует доступ пользователя к рабочему компьютеру (или же производит другие действия, необходимые для обеспечения безопасности информации). Развернутая нейронная сеть для корректной и точной работы должна заново обучаться / дообучаться с некоторой периодичностью (например, раз в сутки), чтобы постоянно находиться в актуальном состоянии (так как в данном случае “concept drift” происходит довольно часто – пользователи постоянно немного меняют скорость печати и корректировка раз в сутки поможет успешно справиться с повышением качества распознавания нейронной сети). Это дообучение должно проходить довольно быстро, чтобы качество системы всегда находилось на должном уровне.

Сервис с развернутой моделью нейронной сети для обучения модели, отправляет задание на обучение на высокопроизводительный кластер. Его же будем считать и системой отслеживания заданий обучения нейронных сетей на кластере, как инициатором этого процесса. Также сервис отправляет набор накопленных данных. Набор отправляемых данных для обучения модели может быть достаточно велик (в зависимости от решаемой задачи. Например, обучение speech-to-text моделей требует, как минимум, гигабайтов данных и большого числа эпох). Чтобы убедиться, что кластер получит доступ ко всем данным – необходимо отправить данные в некоторое внешнее хранилище, например объектное облачное хранилище S3. Скрипт, исполняемый на кластере, может получить доступ как ко всему набору данных в любой момент времени из внешнего облачного хранилища, или же получать данные по частям.

После того, как сервис с развернутой моделью получает уведомление, что данные были успешно загружены во внешнее хранилище (или убеждается в этом, периодично опрашивая хранилище и сверяя контрольную сумму), можно запускать обучение модели глубокого обучения на кластере. Задание, отправляемое на кластер, может не сразу попасть на исполнение (зависит от реализации внешнего интерфейса кластера). Прежде оно попадает в очередь заданий кластера.

Состояние выполнения заданий отслеживается с некоторой периодичностью (или получать уведомления об изменениях в кластерной среде). Необходимо учесть вероятность потери пакетов о состоянии. Поэтому запрос состояния несколько раз дублируется в случае отсутствия ответа.

Заданиям, которые планируется отправить на кластер присваивается некоторый приоритет.

Получив уведомление и информацию о состоянии загрузки ресурсов кластера, а также текущих задачах в очереди и на вычислении – система определяет следующее отправляемое вычислительное задание (на основе очереди задач, приоритетов задач, а также состояния кластерной среды и загрузки ресурсов кластера). В таком случае также следует принимать во внимание алгоритм распределения ресурсов между процессами, а также алгоритм управления запуском заданий для кластерных систем. [3]

В случае, если задание на «обучение» данной нейронной сети уже имеется на кластере, но были получены более новые данные, на которых следует обучить нейронную сеть, то возможны два варианта. Либо старое задание снимается с выполнения и заменяется новым. Либо же в течение некоторого регламентного времени происходит ожидание окончание выполнения задания и в случае, если оно превышено – происходит замена задания. В данном случае задание заменялось сразу.

Помимо контроля очередей задач, также необходима система планирования вычислений. Обычно она зависит от типа кластера и реализуется на его стороне. Существующие системы, реализующие планирование вычислений, подразделяются на следующие группы: кластерные, списочные, генетические и смешанные. [4]

Благодаря использованию планировщика задач был получен значительный выигрыш в скорости дообучения нейронных сетей (так как в случае отсутствия планировщика, задачи на дообучение поступали бесконтрольно и в случае отсутствия доступных ресурсов или наличия подобной задачи на кластере – время получения результата значительно увеличивалось по сравнению с контролем задач, в том числе дублей, и ресурсов планировщиком)

Рисунок 1. Сравнение скорости вычисления задачи, начиная отсчет от необходимости вычисления

Далее скрипт вычислений запускается на кластере. Планировщику задач также необходимо удостовериться, что скрипт попал в очередь вычислений на кластере.

Сам скрипт, реализующий обучение нейронной сети на кластере, также следует реализовывать с учетом параллельной и распределенной архитектуры кластера. Необходимо учитывать межпроцессорные взаимодействия, а также распределить вычисления так, чтобы каждый процессор был занят решением своей части задачи. Также желательно, чтобы как можно меньший объем данных пересылался между процессорами, поскольку такие коммуникации значительно более медленные операции (а в случае распределенных вычислений еще и зависят от задержки сети), чем сами вычисления. [5]

По окончанию вычислений – скрипт-задача, отправляемая на кластер, сформировывает некоторый результат, полученный в результате вычислений. В рассматриваемом случае это сериализованная модель глубокого обучения, а также набор «чекпоинтов», с которых можно восстановить модель в любой среде, имеющей необходимое окружение. Этот результат отправляется на внешнее хранилище.

Система, занимающаяся отправкой, управлением и контролем задач получает уведомление (или, опять же, в ходе проверок обнаружить необходимый результат) о наличии результата, полученного по окончанию вычисления задачи на кластере. Затем необходимо выгрузить результат на сервис и развернуть его для дальнейшего исполнения.

Диаграмма последовательностей взаимодействия системы и кластера проиллюстрирована ниже:

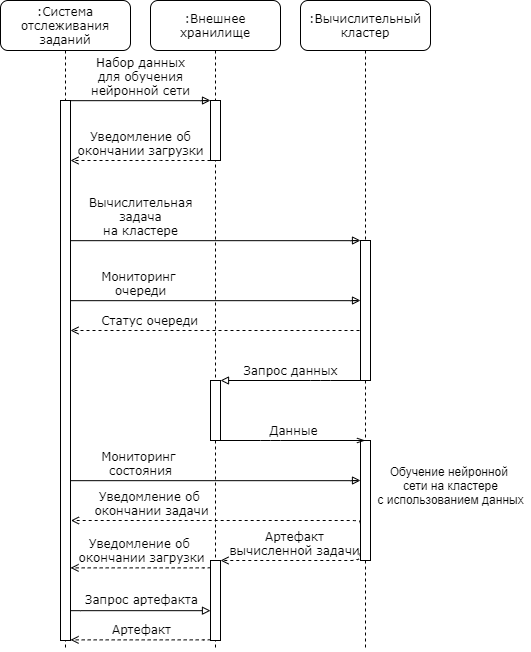


Рисунок 2. Диаграмма последовательностей. Взаимодействие системы и кластера

**Заключение**

Применение системой отслеживания заданий необходимо для любой системы, осуществляющей взаимодействие с некоторой внешней средой и отправляющее в нее задания на вычисления.

При использовании других вычислительно-сложных задач (отличных от «обучения» нейронных сетей, но также имеющих большое количество математических вычислений) также применим данный метод и система для взаимодействия с кластером.

Описанный в статье планировщик задач подходит ко многим реализациям очередей в кластерных средах. Однако для него необходимо наличие внешнего интерфейса и методов проверки статуса очереди на кластере, ресурсов на кластере, статуса задач, выполняемых на кластере. В случае отсутствия данных интерфейсов и методов – необходимо будет использовать другие методы.

**Список литературы**

1. Аль-хулайди Абдулмаджид Ахмед Галеб (2011) – «Распределенные вычисления (кластерные вычисления) с использованием пакета параллельного программирования»;
2. Ограничения машинного обучения в NLP/NLU: сайт. URL: <https://aikho.github.io/2018/06/24/limitations-of-ml-for-nlp.html> (дата обращения: 26.04.2020);
3. А.А. Аль-Хулайди, Ю.О. Чернышёв (2011) – «Разработка алгоритмов управления заданиями при организации параллельных вычислений в кластерных вычислительных системах»;
4. О.В. Русанова, Ярох, Ю.А. (2011) – «Планирование вычислений в гетерогенных кластерных системах»;
5. Т.Л. Тен, Г.Д. Когай, Н.И. Томилова (2014) – «Кластерные системы для организации параллельных вычислений»
6. В.Г. Богданова, И.В. Бычков, А.С. Корсуков, Г.А. Опарин, А.Г. Феоктистов (2014) – «Мультиагентный подход к управлению распределенными вычислениями в кластерной GRID – системе»
7. А.Н. Горбунов, Т.В. Лелетко (2014) – «Компьютерная модель кластера параллельных вычислений»
8. В.Д. Павленко, В.В. Бурдейный – «Технология программирования вычислений в кластерных системах с использованием транспарентного распараллеливания, основанного на заказах»

Ковтушенко Александр Петрович, доцент кафедры «Программное обеспечение ЭВМ и информационные технологии», МГТУ им. Н. Э. Баумана, Москва, Российская Федерация

Новоженов Владимир Алексеевич – магистр, студент кафедры «Программное обеспечение ЭВМ и информационные технологии», МГТУ им. Н. Э. Баумана, Москва, Российская Федерация