О. М. Олиферович

ПАРАЛЛЕЛИЗМ ДАННЫХ И МОДЕЛЕЙ ДЛЯ ГЛУБОКИХ НЕЙРОННЫХ СЕТЕЙ

В работе приводится описание распараллеливания обучения глубоких нейронных сетей с использованием параллелизма данных или моделей.

Введение

Существующие системы глубокого обучения обычно распараллеливают обучение глубоких нейронных сетей (DNN – Deep Neural Networks) с использованием параллелизма данных или моделей, но эти стратегии часто приводят к неоптимальной производительности распараллеливания [1]. По мере развития методов глубокого обучения, DNN-модели стали значительно крупнее и дороже в вычислительном отношении для обучения. В результате в настоящее время существует стандартная практика распараллеливания обучения DNN между распределенными гетерогенными кластерами. Хотя модели DNN и кластеры, используемые для их распараллеливания, становятся все более сложными, стратегии, используемые современными системами глубокого обучения (например, TensorFlow, Caffe2 и MXNet) для распараллеливания обучения, остаются простыми и часто неоптимальными.

I. Стратегии распараллеливания

Одной из наиболее распространенных стратегий распараллеливания является параллелизм данных, который помещает реплику всей нейронной сети на каждое устройство, так что каждое устройство обрабатывает подмножество обучающих данных и синхронизирует параметры сети между репликами в конце итерации. Параллелизм данных эффективен для интенсивных вычислений операторов с несколькими обучаемыми параметрами.

Другой распространенной стратегией распараллеливания является модельный параллелизм, который разбивает DNN на дизъюнктивные подмножества и обучает каждое подмножество на выделенном устройстве, что снижает затраты на связь для синхронизации параметров сети, но предоставляет ограниченный параллелизм.

Разработанные экспертами стратегии распараллеливания вручную оптимизируют распараллеливание для конкретных DNN, используя знания и опыт экспертов в предметной области. Для распараллеливания используется параллелизм данных, который копирует данные на каждом узле и переключается на модель параллелизма для распараллеливания внутри узла. Хотя эти стратегии, повышают производительность по сравнению с параллелизмом данных и моделей, они являются неоптимальными.

II. SOAP пространство поиска стратегий распараллеливания

Используя комплексное SOAP (Sample-Operator-Attribute-Parameter) пространство поиска стратегий распараллеливания для DNNs, которое обобщает и выходит за рамки предыдущих подходов, можно получить улучшение результатов обучения. Размер оператора описывает, как распараллеливаются различные операторы в DNN. Для одного оператора размеры выборки и параметра указывают, как распределяются обучающие выборки и параметры модели между устройствами. Наконец, измерение атрибутов определяет, как секционируются различные атрибуты в выборке (например, размеры высоты и ширины изображения).

Существующие подходы рассматривают только одно или подмножество измерений SOAP. Например, параллелизм данных использует размерность выборки для распараллеливания обучения, в то время как параллелизм модели использует размерность параметра и оператора. Разработанные экспертами стратегии используют параллелизм в измерении выборки или параметра для распараллеливания оператора, но не поддерживают гибридный параллелизм.

III. FlexFlow

FlexFlow рассматривает возможность распараллеливания любой DNN (линейной или нелинейной) во всех измерениях SOAP и исследует более полное пространство поиска, включающее существующие подходы в качестве частных случаев [2]. В результате FlexFlow способен разрабатывать стратегии распараллеливания, которые значительно превосходят существующие подходы.

Ключевая задача, которую FlexFlow должен решить, заключается в том, как эффективно исследовать пространство поиска SOAP, которое намного больше, чем те, которые рассматривались в предыдущих системах, и включает в себя более сложные стратегии распараллеливания. С этой целью FlexFlow использует два основных компонента: быстрый инкрементный симулятор выполнения для оценки различных стратегий распараллеливания и алгоритм поиска по цепочке Маркова Монте-Карло (MCMC) [3], ко-

торый использует преимущества инкрементного симулятора для быстрого исследования большого пространства поиска (см.рис.1.)

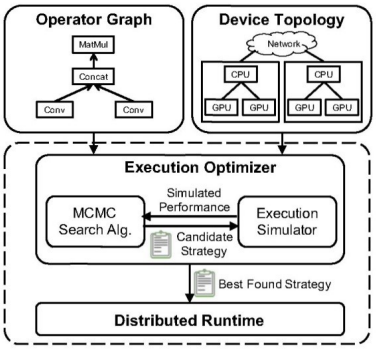

Рис. 1 – Структура FlexFlow

Имитатор выполнения обеспечивает высокую точность прогнозирования производительности распараллеливания в пространстве поиска SOAP для произвольных DNNs и на три порядка быстрее, чем профилирование реальных исполнений. При измерении производительности оператора один раз для каждой конфигурации эти измерения передаются в граф задач, который моделирует как архитектуру модели DNN, так и топологию сети кластера. Имитатор выполнения оценивает производительность стратегии распараллеливания, имитируя выполнение на графике задач.

Подобно существующим фреймворкам глубокого машинного обучения (например, TensorFlow и PyTorch), FlexFlow использует операторный граф Q для описания всех операторов и состояний в DNN. Каждый узел является оператором, и каждое ребро представляет собой тензор. FlexFlow принимает схему топологии графа и топологию устройства в качестве входных данных и автоматически находит эффективную стратегию в пространстве поиска SOAP. Все стратегии в пространстве выполняют одно и то же вычисление, определенное DNN, и поэтому поддерживают одинаковую точность модели по дизайну.

Оптимизатор выполнения использует алгоритм поиска MCMC для исследования пространства возможных стратегий распараллеливания и итерационно предлагает возможные стратегии, которые оцениваются симулятором выполнения. Имитатор выполнения использует алгоритм Дельта-моделирования, который имитирует новую стратегию с использованием инкрементных обновлений к предыдущим компонентам. Когда бюджет времени поиска исчерпан, оптимизатор выполнения отправляет наилучшую обнаруженную стратегию в распределенную среду выполнения для распараллеливания фактических выполнений.

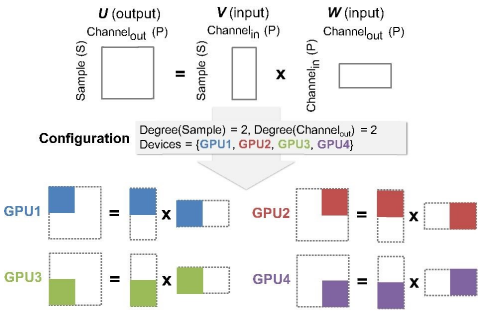

Рис. 2 – Структура FlexFlow

На рисунке 2 показан пример конфигурации распараллеливания для оператора умножения матрицы (т.е. U = VW). Оператор разделен на четыре независимые задачи, назначенные различным устройствам GPU. Граф задач моделирует зависимости между отдельными задачами, производными от операторов. Чтобы унифицировать абстракцию, мы моделируем каждое аппаратное соединение между устройствами как коммуникационное устройство, которое может выполнять только коммуникационные задачи (то есть передачу данных). При сравнении каждой итерации обучения результативность FlexFlow с учетом исходных условий оказалась не хуже, чем у TenzorFlow.

IV. Выводы

Параллелизм данных часто используются в существующих системах глубокого обучения. Для контроля за различиями в реализации, были проведены эксперименты обработки данных в TensorFlow r.1.7 и PyTorch v.0.3. В сравнении с TensorFlow и PyTorch, FlexFlow достигает той же или более высокой производительности вычислений по всем шести DNN.

1. Dean, J., Corrado, G. S., Monga, R., Chen, K., Devin, M., Le, Q. V., Mao, M. Z., Ranzato, M., Senior, A., Tucker, P., Yang, K., and Ng, A. Y. Large scale distributed deep networks. In NIPS, 2012.
2. Krizhevsky, A., Sutskever, I., and Hinton, G. E. ImageNet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems, NIPS, 2012.
3. Hastings, W. K. Monte carlo sampling methods using markov chains and their applications. Biometrika, 57 (1):97–109, 1970.

Олиферович Ольга Михайловна, студентка кафедры информационных технологий и управления БГУИР, olya.oliferovich@mail.ru.

Научный руководитель: Трофимович Алексей Федорович, старший преподаватель кафедры информационных технологий и управления БГУИР, trofimaf@bsuir.by.