Презентация к предзащите дипломной работы  
Тема: "Аннотирование промежуточного представления MLIR на основе профиля исполнения"

Слайд 1: Заголовок  
Здравствуйте, меня зовут Алексеев Алексей, и моя дипломная работа посвящена созданию системы аннотирования промежуточного представления (IR) компилятора MLIR на основе профиля исполнения.

Слайд 2: Актуальность и мотивация

Современные программы машинного обучения получили огромный рост в сложности, объемах обрабатываемой и хранимой информации. Такие изменения напрямую связаны с методом их улучшения – экспонциальное увеличение числа параметров, весов модели. Очевидно, нужно каким то образом обеспечить инструментально такую быстро растущую сложность. Одним из таких инструментов стал MLIR, представленный в рамках проекта LLVM в 2019 году.

Слайд 3: Что такое MLIR и зачем он нужен  
MLIR — это инфраструктура для построения компиляторов с поддержкой многоуровневого промежуточного представления. Разница в уровнях абстракций компилируемых языков c, c++ от интерпретированных, в нашем случае это широко используемый для написания программ машинного обучения – python велика и в частности выражена в особенности используемых типов. Возможность построения собственных промежуточных представлений или часто диалектов призвана закрыть этот разрыв абстракций и предоставить возможность разработки оптимизирующих проходов (далее pass-ов) оперирующих в терминах тензоров, многомерных сверток и операций линейной алгебры.

Слайд 4: Цель — использовать профиль в компиляции

Профиль это набор численных, структурных характеристик, отражающих особенность исполнения программы на конкретном устройстве (CPU, NPU, GPU). Другими словами это пульс программы, ритм которого можно использовать в качестве обратной связи при компиляции. Все существующие на сегодняшний день решения написанные на основе MLIR используют эвристики ничего не знающие об особенностях архитектуры конкретного устройства. Опыт оптимизаций определяемых профилем исполнения (PGO) в рамках LLVM, создает оптимистичное веяние для повторение данного подхода в рамках MLIR. Целью данной работы является создание инструмента автоматического аннотирования временем исполнения операций диалекта промежуточного представления MLIR.

Слайд 5: Постановка задач

Существует несколько основных фреймворков, представляющих инструментарий разработки программ машинного обучения – TensofFlow, PyTorhch, Onnix. В рамках собственных библиотек они создали пользовательские высокоуровневые диалекты, доступные для сторонних расширений и написания pass-ов оптимизаций. Вместе с этим, они предоставляют готовые решения профилировщиков, способные собирать разнообразную системную информацию времени исполнения. Для разработки трансформаций на аннотированном графе операций, бывает полезно получить визуальное представление численных и структурных данных профиля.

Из существующих требований естественно вытекают задачи:

* Расширение диалекта TensorFlow полями аннотаций о времени начала, выполнения операций.
* Сериализация выходного формата данных профиля TensorFlow в универсальное представление в виде структуры граф.
* Инструмент визуализации сериализованного графа для обнаружения узких мест.

Слайд 6: Архитектура системы

Архитектура итогового решения имеет вид:

1. Сбор профиля времени исполнения операций
2. Построение графа и визуальный анализ узких мест
3. Аннотирование данными сериализованного графа промежуточного представления в отдельном проходе
4. Применение пассов оптимизаций на основе профиля

Основным преимуществом предложенной архитектуры является дальнейшая расширяемость. Сбор данных профиля может производиться произвольным профилировщиком, тогда как итоговые данные имеют унифицированный формат и беспрепятственно используются при аннотировании далее. Аннотирование реализовано в виде интерфейсных трейтов операций, что позволяет удобно и легко изменять количество, набор операций используемых при анализе. Пассы связанные с PGO вынесены в отдельный pipeline, что позволяет удобно и структурно зарегистрировать оптимизационные проходы на основе добавленных данных профиля.

Слайд 7: Пример входной программы и результата прохода компиляции

Здесь нужно привести два листинга – программы и аннотированного IR. Обсудить детали связанные с lowering. Можно ли сохранить добавленные данные в поля операций.

Слайд 8: Выводы и дальнейшие шаги  
Итак:

* Используются трейты и интерфейсы операций для добавления аннотаций.
* Для хранения данных профилировщика используется унифицированный сериализовынный в json граф исполнения операций.
* Разработан инструмент для визуального анализа.

Следующие шаги:

* Адоптирование решения для работы с данными профиля NPU Ascent
* Разработка CLI-инструмента
* Разработка оптимизирующих проходов на основе добавленных данных профиля

Спасибо за внимание! Готов ответить на вопросы.