



НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. Н.И. ЛОБАЧЕВСКОГО

ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МАТЕМАТИКИ И МЕХАНИКИ

КАФЕДРА ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ И СИСТЕМНОГО ПРОГРАММИРОВАНИЯ

ЛАБОРАТОРИЯ ITLav



НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. Н.И. ЛОБАЧЕВСКОГО
ИНСТИТУТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МАТЕМАТИКИ И МЕХАНИКИ
КАФЕДРА ВЫСОКОПРОИЗВОДИТЕЛЬНЫХ ВЫЧИСЛЕНИЙ И СИСТЕМНОГО ПРОГРАММИРОВАНИЯ
ЛАБОРАТОРИЯ ITLAB

***ВВЕДЕНИЕ В АНАЛИЗ ПРОИЗВОДИТЕЛЬНОСТИ
И ОПТИМИЗАЦИЮ ПРОГРАММ***

**Оптимизация библиотеки CatBoost
для использования векторных
расширений RISC-V**



**LOBACHEVSKY
UNIVERSITY**

Мееров И.Б., Козинов Е.А., Васильев Е.П.,
Горшков А.В., Кустикова В.Д., Маклаев А., Волокитин В.Д.

Содержание

- ❑ Цель
- ❑ Что такое CatBoost?
- ❑ Алгоритм
- ❑ Оптимизация
- ❑ Результаты

Мотивация

❑ Почему мы заинтересовались устройствами RISC-V?

- Свободная, открытая, перспективная архитектура
- Первые устройства RISC-V достаточно легко использовать
- Системное ПО приемлемого качества и постоянно развивается
- Программные библиотеки часто могут быть построены и использованы без модификаций (достижение высокой производительности может быть непростым делом)
- **RISC-V это глоток свежего воздуха!**

❑ Ложка дегтя:

- Не хватает инструментов для анализа производительности
- Не хватает документации и методических статей
- Не хватает устройств HPC-класса

❑ Проблемы нас не останавливают, а мотивируют

Цель

- ❑ **Ключевой вопрос: как оценивать производительность больших программных комплексов и какие техники могут улучшить производительность на устройствах RISC-V?**
 - Рассматриваем **CatBoost**, один из широко распространенных пакетов ML, основанный на деревьях решений
 - Рассматриваем алгоритмы CatBoost как **«черный ящик»**, не погружаясь глубоко в логику алгоритмов (*типично для performance engineering*)
 - Фокусируемся на **аспектах производительности** и ищем пути **ускорить код**, обнаружив и векторизовав основные **вычислительно-трудоемкие циклы** при работе на выбранных наборах данных

Что такое CatBoost?

- ❑ **CatBoost** реализует алгоритм, основанный на **градиентном бустинге деревьев решений**, пакет разработан **Yandex**
- ❑ Используется в *поисковых и рекомендательных системах*, персональных помощниках, ПО для автомобилей и др., в частности, в Yandex, CERN...
- ❑ Библиотека с **открытым исходным кодом** (Apache 2.0)
- ❑ Решает задачи классификации, регрессии и др.
- ❑ Python, R, Java и C++ APIs
- ❑ Оптимизирован для вычислений на CPU и GPU
- ❑ Включает средства для тренировки и вывода моделей, анализа качества и визуализации

Градиентный бустинг деревьев решений

- ❑ CatBoost реализует градиентный бустинг деревьев решений
- ❑ **Идея бустинга** – построить сильные предсказательные модели, используя ансамбли слабых моделей
- ❑ Алгоритм использует «забывчивые деревья решений» (*oblivious decision trees*) с малой глубиной в качестве слабых моделей
- ❑ **Забывчивое дерево** – упрощенная модель дерева решений, где все узлы на одном уровне проверяют одно и то же условие
- ❑ В ходе тренировки, деревья добавляются к ансамблю, и каждое дерево пытается исправить ошибки предыдущего

Алгоритм CatBoost

Пример:
задача классификации

Вход



2D AoS (RGB)

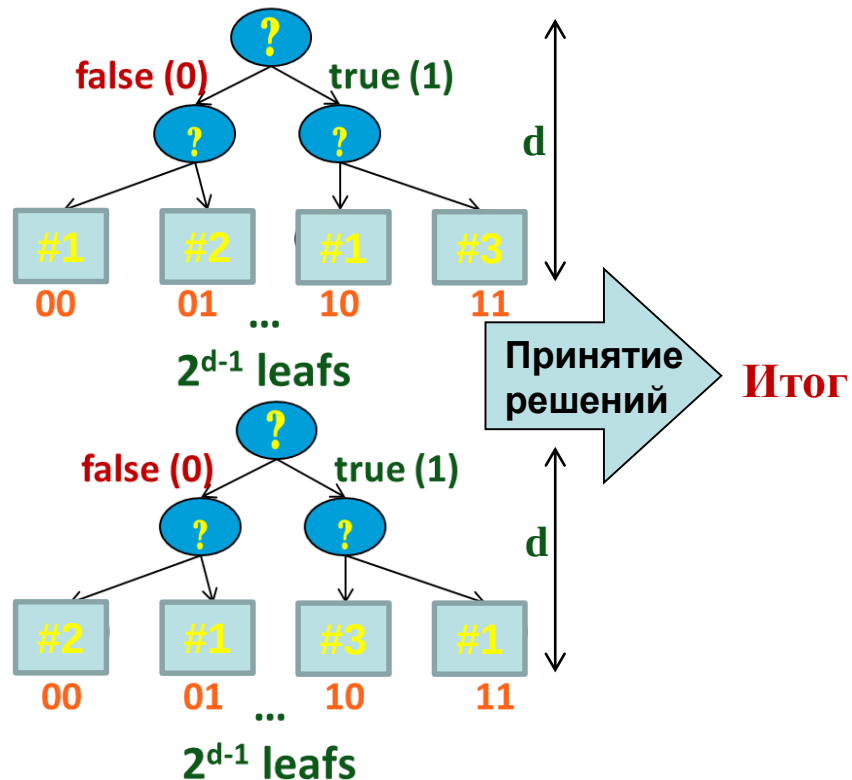
Выделение признаков

Вектор признаков



1D fp массив

Ансамбль деревьев

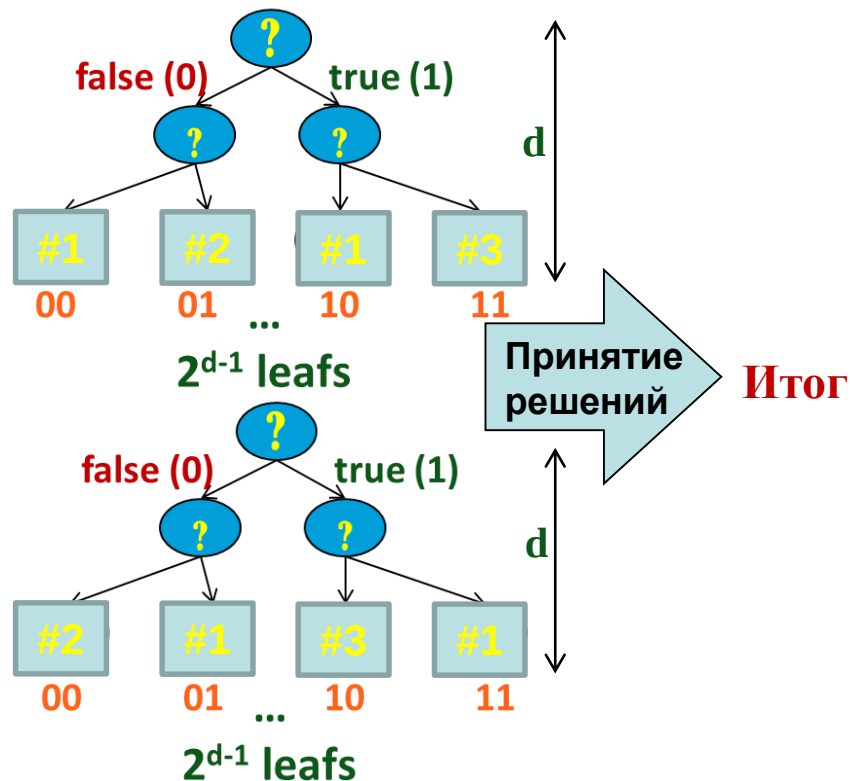


Алгоритм CatBoost

Пример:
задача классификации

- ❑ Забывчивое дерево это упрощенное дерево (на одном уровне все узлы содержат одно и то же условие)
- ❑ Оптимизация (реализовано Yandex)
 - не нужно хранить деревья напрямую
 - не нужны традиционные проверки
- ❑ ! Достаточно отсортировать все натренированные пороги, сделать бинаризацию и использовать битовые операции, чтобы найти результирующий лист дерева
- ❑ Наша цель: улучшить производительность на RISC-V CPUs, не меняя алгоритм и структуры данных

Ансамбль деревьев



Наборы данных

- Используем несколько распространенных наборов данных
- Нет гарантии, что набор *репрезентативен*, но, по крайней мере, мы покрываем несколько разных сценариев

Набор данных	Строки/Столбцы	Число классов	Функция потерь	Темп обучения	Глубина дерева
MQ2008	9630 x 46	-	YetiRank	0.02	6
Santander customer transaction	400k x 202	2	LogLoss	0.01	1
Covertypes	464.8k x 54	7	MultiClass	0.50	8
YearPredictionMSD	515k x 90	-	MAE	0.30	6
image-embeddings	5649 x 512	20	MultiClass	0.05	4

Наборы данных (краткое описание)

❑ “Covertypes”

- 52 целочисленных и бинарных признака, представляющих дикие местности и типы почв
- требуется прогнозировать тип лесного покрова (7 классов)
- набор данных был разделен на обучающую и тестовую выборку в соотношении 70:30

❑ “Santander customer transaction”

- 200 ненормализованных признаков, бинарная классификация
- тренировочная и тестовая выборки имеют по 200 000 элементов каждая

❑ “YearPredictionMSD”

- 90 ненормализованных признаков, извлеченных из песен
- требуется прогнозировать год выхода песни
- 463 715 сэмплов в тренировочной выборке и 51 630 в тестовой

Наборы данных (краткое описание)

❑ “MQ2008”

- 46 признаков для решения задачи ранжирования
- содержит 9 630 тренировочных сэмплов и 2 874 тестовых сэмплов

❑ “image-embeddings”

- подмножество набора данных PASCAL VOC 2007 (20 классов)
- не содержит изображений с объектами нескольких классов
- содержит 9 630 тренировочных сэмплов и 2 874 тестовых сэмплов
- эмбединги генерируются для изображений с использованием натренированной модели resnet34 из библиотеки TorchVision
- для получения эмбедингов, из модели убран последний классификационный слой

Возможности для оптимизации

- ❑ **x86 CPUs:** множество средств для анализа и оптимизации оптимизации (VTune, Advisor, компиляторы C, C++, Fortran...)
- ❑ **RISC-V CPUs:**
 - Компиляторы на ранней стадии разработки (весьма неплохого качества, тем не менее)
 - Набор средств для оптимизации существенно ограничен
 - Их функциональность скудная (могут лишь искать хотспоты)
- ❑ **Как оптимизировать большие коды?** (*стандартно для perf. eng.*)
 - Найти хотспоты (вычислительно-трудоемкие участки кода)
 - Попробовать векторизовать вычислительные циклы
 - Проверить и улучшить эффективность распараллеливания
- ❑ **Что мы сделали:** вручную векторизовали код на **RVV 0.7.1**

Как найти хотспоты на устройствах RISC-V?

- ❑ Выяснилось, что использовать ***perf*** из Linux для анализа CatBoost **проблематично**, т.к. код содержит *интерфейс* на Python и *динамические библиотеки* на C++
- ❑ **Технология профилирования:**
 - Найти точку входа – наиболее нагруженную функцию (**perf**)
 - Изучить тело функции, вставить замеры времени и аккумулировать времена в глобальную переменную (**свой таймер**)

https://github.com/itlab-vision/catboost/blob/catboost_1_2_2_rvv/catboost/time_profile.h

- Проанализировать результаты, продвинуться вглубь, построить граф вызова функций

Как найти хотспоты на устройствах RISC-V?

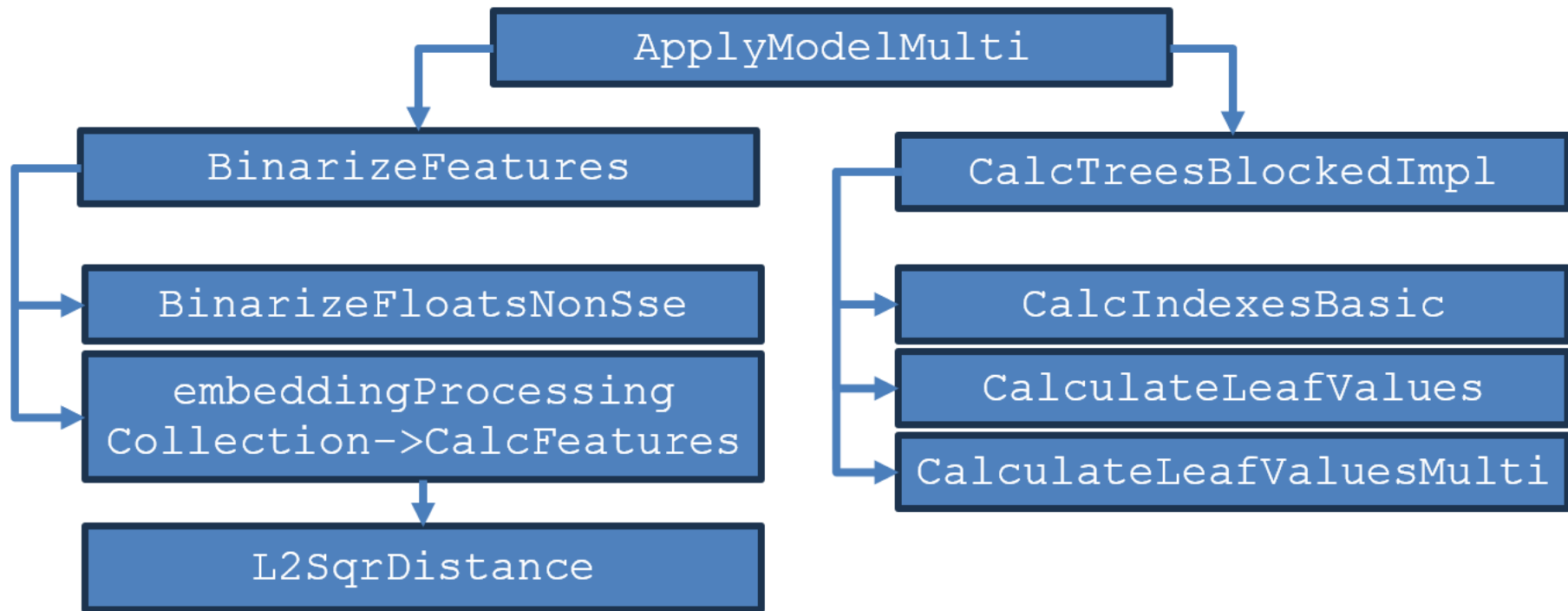
❑ Технология профилирования:

- Найти точку входа – наиболее нагруженную функцию (**perf**)
- Изучить тело функции, вставить замеры времени и аккумулировать времена в глобальную переменную (**свой таймер**)
- Проанализировать результаты, продвинуться вглубь, построить граф вызова функций

❑ Валидация:

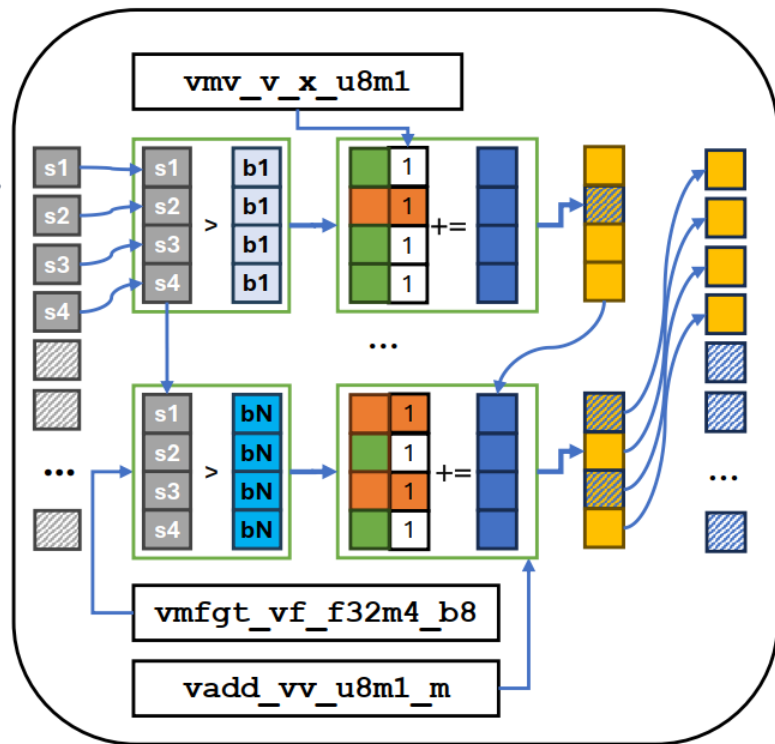
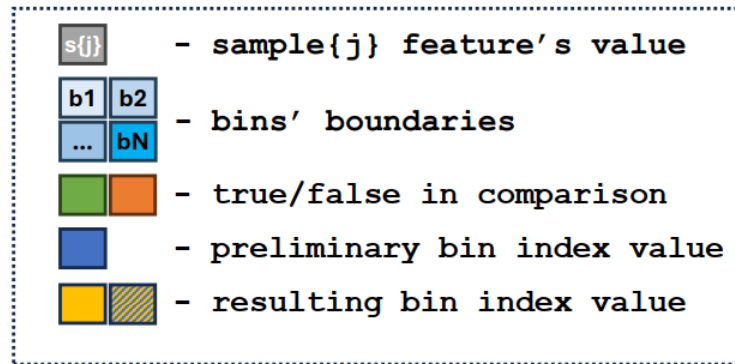
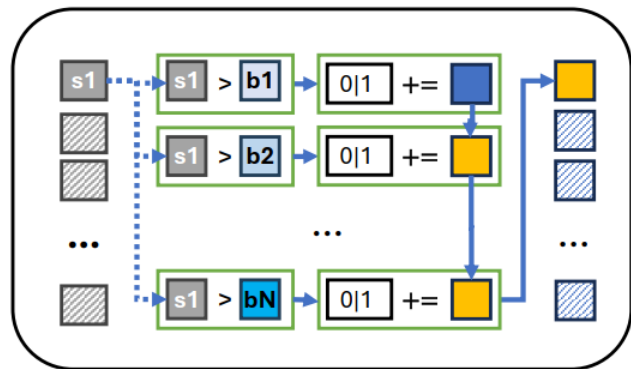
- Использовать тот же подход на x86 CPUs и сравнить с результатами профилировки имеющимися на x86 средствами
- Сравнить запуски с профилировкой и без нее на RISC-V CPUs
- Провести **итоговые эксперименты без профилировки на полных наборах данных**

Кандидаты для векторизации



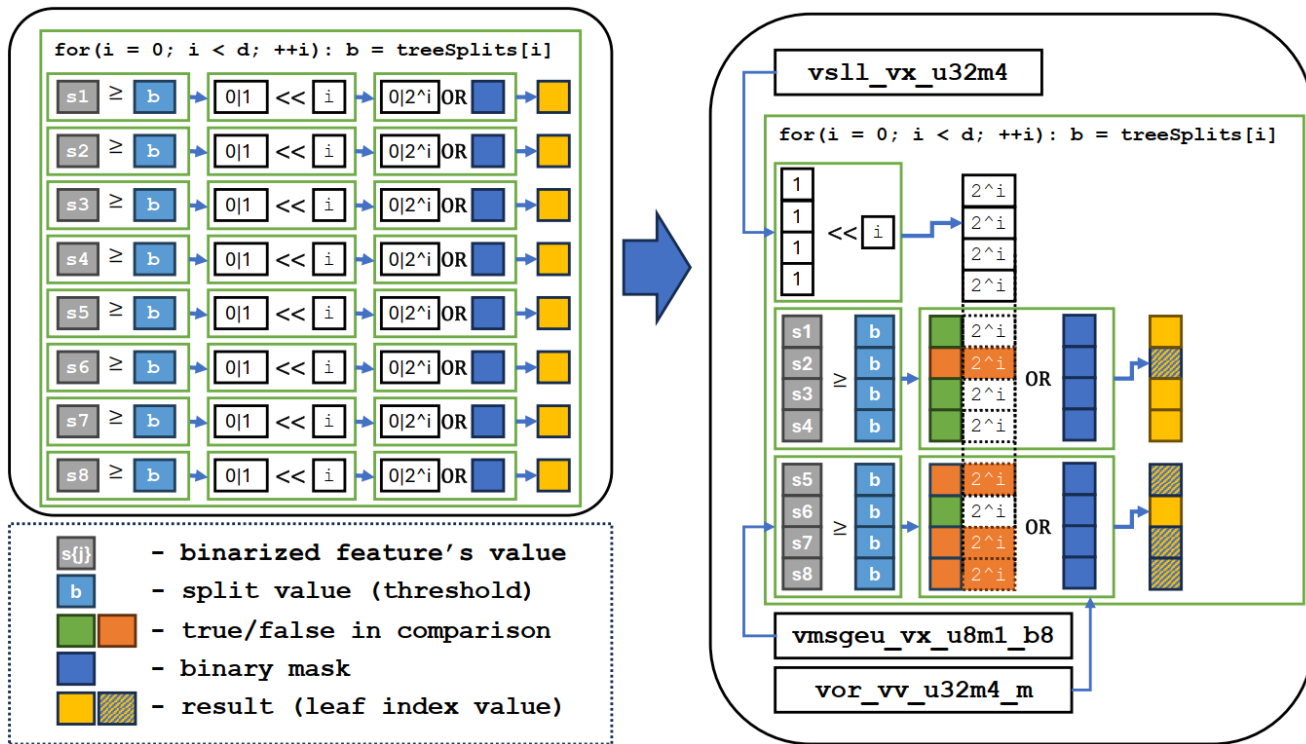
Функция BinarizeFloatsNonSse()

□ Программная реализация



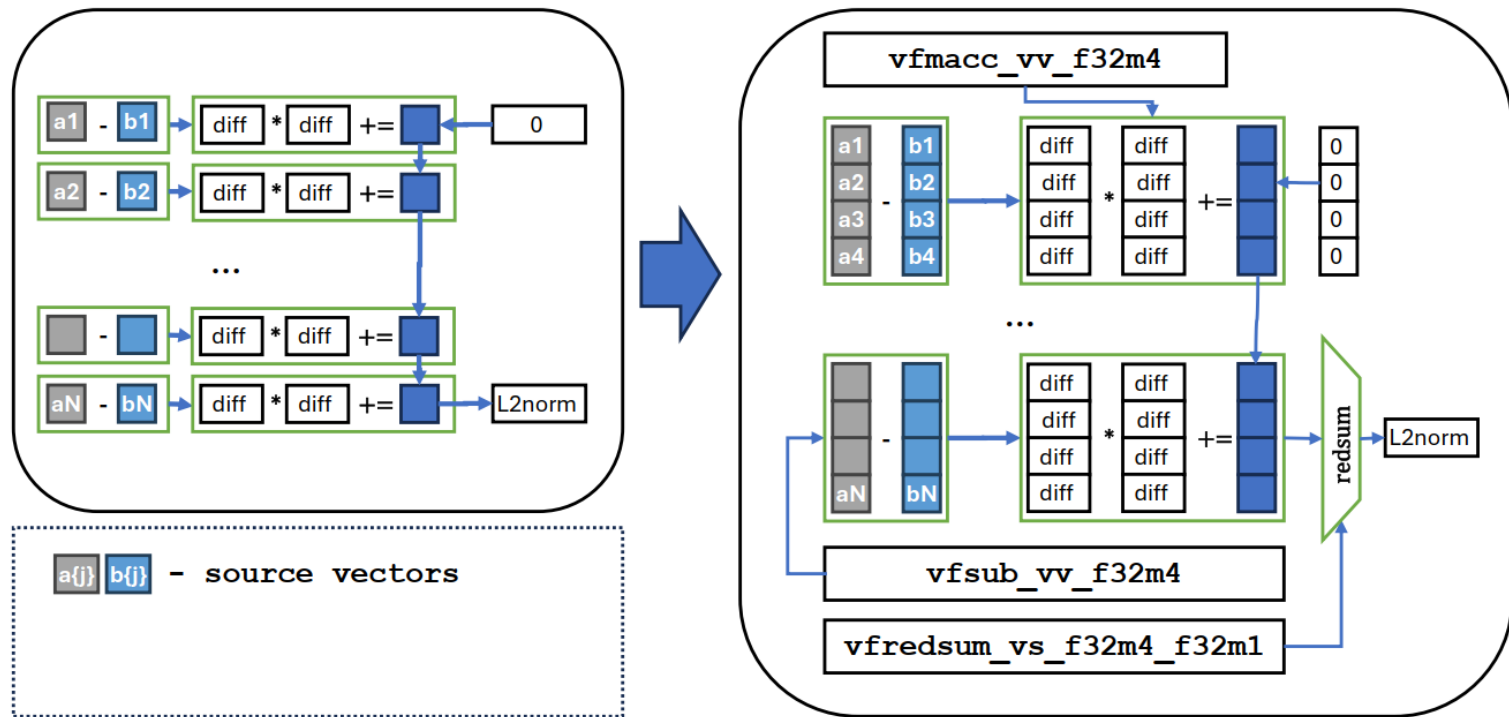
Функция CalcIndexesBasic()

Программная реализация



Функция L2SqrDistance()

Программная реализация



Векторизация для RISC-V CPUs: Идея

- ❑ Векторное расширение RVV 0.7.1 позволяет *выбрать число векторных регистров, используемых в одной операции*
- ❑ Например, мы можем использовать **блок из 4х 128-битных регистров** и соответствующие векторные операции, как будто архитектура содержит **512-битные регистры**
- ❑ Для этого нужно указать суффикс **m4** в наименованиях интринсиков
- ❑ Таким образом можно значительно повысить производительность, но определить наилучший вариант (m1, m2, m4, m8) пока можно только экспериментально
- ❑ **Мы использовали наилучший вариант**

Методика проведения вычислительных экспериментов

- ❑ Тестирование производительности и корректности можно провести, сравнив результаты работы CatBoost на процессорах x86 (**код оптимизирован разработчиками CatBoost**) и процессорах RISCV (**базовый скалярный код от разработчиков CatBoost и наша векторизованная реализация**).
- ❑ Мы разделили все эксперименты на две группы:
 - Первая группа экспериментов использовалась для первоначальной оценки корректности и производительности во время разработки.
 - Проводилась на *сокращенных наборах данных* (1000 сэмплов).
 - Все запуски выполнялись в *последовательном режиме*.
 - Вторая группа экспериментов включала **масштабные запуски в многопоточном режиме** на **полных наборах данных** и использовалась для окончательной оценки.

Оборудование

- ❑ **X86** (тренировка моделей, подготовка наборов данных, вычисление точности, сравнение производительности):
 - Intel Xeon Silver 4310T (2 CPUs с 10 ядрами, 20 ядер всего), 64 GB RAM
- ❑ **RISC-V: Mini-cluster Lichee Cluster 4A**
 - 7 план с RISC-V TH1520 CPUs на базе ядер C910
 - Каждый процессор содержит 4 ядра с поддержкой RVV 0.7.1
 - Каждая плата имеет 16GB RAM



Результаты вычислительных экспериментов.

Первая группа экспериментов. (1)

Результаты профилирования CatBoost на наборе данных YearPredictionMSD на процессоре RISC-V. Код выполнялся в последовательном режиме. Время указано в с.

Function/metric	Call count	Baseline		Optimized		Speedup
		time	% total time	time	% total time	
CalcTreesBlockedImpl	8	1.35	89.41%	0.39	79.82%	3.43
CalcIndexesBasic	79992	1.02	67.60%	0.07	15.11%	13.68
CalculateLeafValues	79992	0.21	13.70%	0.20	40.56%	1.03
BinarizeFloatsNonSse	720	0.09	5.63%	0.03	6.05%	2.85
Other (profiler, auxiliary func ...)		0.07	4.95%	0.07	14.14%	-
Total time		1.51		0.49		3.06

Результаты вычислительных экспериментов.

Первая группа экспериментов. (2)

Результаты профилирования CatBoost на наборе данных **covertype** на процессоре RISC-V. Код выполнялся в последовательном режиме. Время указано в с.

Function/metric	Call count	Baseline		Optimized		Speedup
		time	% total time	time	% total time	
CalcTreesBlockedImpl	8	1.45	95.70%	0.81	93.17%	1.79
CalcIndexesBasic	39520	0.70	46.40%	0.06	6.33%	12.76
CalculateLeafValues Multi	39520	0.67	44.44%	0.69	78.64%	0.98
BinarizeFloatsNonSse	432	0.02	1.24%	0.01	1.49%	1.45
Other (profiler, auxiliary func ...)		0.05	3.05%	0.05	5.34%	-
Total time		1.52		0.87		1.74

Результаты вычислительных экспериментов.

Первая группа экспериментов. (3)

Результаты профилирования вывода CatBoost на наборе данных **image-embedding** на процессоре RISC-V. Код выполнялся в последовательном режиме. Время указано в с.

Function/metric	Call count	Baseline		Optimized		Speedup
		time	% total time	time	% total time	
CalcTreesBlockedImpl	8	1.60	8.15%	1.17	18.54%	1.36
CalcIndexesBasic	38064	0.35	1.76%	0.04	0.56%	9.72
CalculateLeafValues Multi	38064	1.18	6.05%	1.07	16.95%	1,11
BinarizeFeatures	1	17.93	91.60%	5.10	80.70%	3.51
BinarizeFloats	312	0.03	0.13%	0.00	0.07%	5.44
NonSse embeddingProcessing Collection		17.91	91.48%	5.10	80.63%	3.51
Other (profiler, auxiliary func ...)		0.05	0.25%	0.05	0.76%	-
Total time		19.58		6.33		3.10

Результаты на полных наборах данных.

Вторая группа экспериментов.

Итоговые результаты сравнения. Код был запущен в многопоточном режиме. Время указано в секундах. Точность одинакова во всех запусках, поэтому отображается только один раз для каждого набора данных.

DataSet	Accuracy	Time (x86)	Time (RISC-V) Baseline	Time (RISC-V) Optimized	Speedup
Santander customer transaction	0.911	0.17	16.07	7.65	2.10
Covertypes	0.960	0.42	59.41	30.60	1.94
YearPredictionMSD	9.168	0.06	16.30	2.79	5.84
MQ2008	0.850	0.02	0.55	0.50	1.10
image-embeddings	0.802	0.18	16.66	6.00	2.78



Выводы

- ❑ Время на сервере x86 приведено только для информации, детальное сравнение с x86 пока не имеет большого смысла
- ❑ Получено значительное (до 6х) ускорение
- ❑ **Два возможных ограничения:**
 - Ускорение достигается при обработке «пачки» входных данных (на одном сэмпле ускорение не ожидается)
 - Хотспоты и распределение нагрузки зависят от **набора данных и решаемой задачи**. Мы исследовали различные модели, но в некоторых других сценариях может потребоваться **дополнительная оптимизация** для эффективного использования ресурсов RISC-V CPU

Литература

❑ Код:

https://github.com/itlab-vision/catboost/tree/catboost_1_2_2_rvv

❑ Статья:

Kozinov, E. *et al.* (2025). Vectorization of Gradient Boosting of Decision Trees Prediction in the CatBoost Library for RISC-V Processors. In: Wyrzykowski, R., Dongarra, J., Deelman, E., Karczewski, K. (eds) Parallel Processing and Applied Mathematics. PPAM 2024. Lecture Notes in Computer Science, vol 15580. Springer, Cham. https://doi.org/10.1007/978-3-031-85700-3_3

Контакты

Нижегородский государственный университет

<http://www.unn.ru>

Институт информационных технологий, математики и механики

<http://www.itmm.unn.ru>

Кафедра высокопроизводительных вычислений и системного программирования

