


Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

Суперкомпьютерные дни в России
2024

Векторизация в библиотеке CatBoost для процессоров RISC-V

23-24 сентября 2024

Евгения Козинов, Евгений Васильев,
Андрей Горшков Валентина Кустикова,
Артем Маклаев, Валентин Волокитин,
Иосиф Мееров

Содержание

- HPC лаборатория
- Мотивация
- Что такое CatBoost?
- Алгоритм
- Улучшение производительности
- Оценка производительности
- Текущие исследования

Изложение ведется по работам:
Kozinov E., Vasiliev E., Gorshkov A., Kustikova V., Maklaev A., Volokitin V., Meyerov I. (2024). Vectorization of Gradient Boosting of Decision Trees Prediction in the CatBoost Library for RISC-V Processors. arXiv preprint arXiv:2405.11062.

Наша HPC лаборатория

Область исследований: разработка научного программного обеспечения, анализ производительности и оптимизация

- Вычислительная физика (лазерная физика, квантовая динамика…)
- Вычислительная биология
- Финансовая математика
- Компьютерная графика, компьютерное зрение, машинное обучение и глубокое обучение
- Графовые алгоритмы, разреженная алгебра

Лаборатория в основном сосредоточена на процессорах x86-64 и программном обеспечении (C, C++, SYCL…)

Какова наша мотивация рассматривать устройства RISC-V?

Мотивация

- Что мотивирует нас рассматривать устройства RISC-V?
  - Архитектура свободна, открыта и очень перспективна
  - Первые устройства RISC-V довольно просты в использовании
  - Стек системного программного обеспечения приемлем и постоянно совершенствуется
  - Библиотеки/инструменты программного обеспечения часто можно скомпилировать и запустить без изменений (достижение высокой производительности может быть сложной задачей)
- **RISC-V — это глоток свежего воздуха!**
- **Ложка дегтя:**
  - Отсутствие инструментов для анализа производительности
  - Отсутствие документации и методических статей
  - Отсутствие устройств, готовых для HPC
- **Но остановило ли это нас?**

Цель

- Главный вопрос — как оценить производительность больших программных пакетов и какие методы применимы для повышения производительности на процессорах RISC-V?
  - Мы рассматриваем **CatBoost**, одну из часто используемых библиотек для решения задач машинного обучения (МО) с использованием деревьев решений в качестве испытательного стенда
  - Мы рассматриваем алгоритмы CatBoost как **«черный ящик»**, не вникая слишком глубоко в логику алгоритмов
  - Мы фокусируемся на аспектах производительности и ищем способы ускорения кода путем **обнаружения** и **векторизации** вычислительно интенсивных циклов вдоль критических путей выполнения программы для различных рабочих нагрузок

Обзор CatBoost

- **CatBoost** реализует алгоритм ML (**градиентный бустинг деревьев решений**), разработанный **Яндексом**
- Используется в *системах поиска и рекомендаций*, персональных помощниках, программном обеспечении для беспилотных автомобилей и во многих других приложениях, в частности, в Яндексе, ЦЕРНе и т. д.
- Доступен как библиотека с *открытым исходным кодом* (Apache 2.0)
- Поддерживает ранжирование, классификацию, регрессию и другие задачи машинного обучения
- Библиотека имеет API на языках Python, R, Java и C++
- Оптимизирован для стандартных вычислений на CPU и GPU
- Включает инструменты для обучения и применения обученных моделей, анализа качества, а также инструменты визуализации

Градиентный бустинг деревьев решений

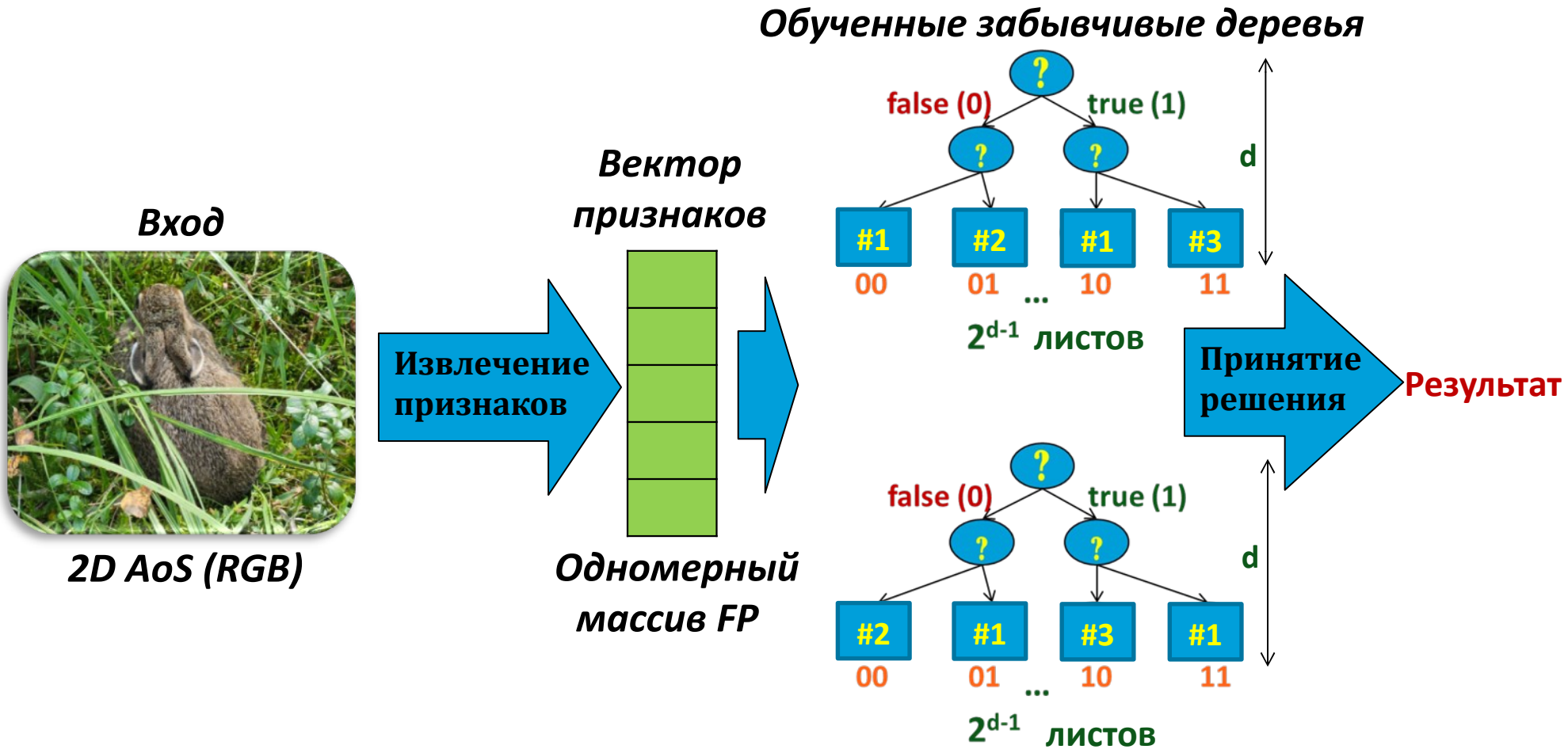
- **CatBoost** реализует *градиентный бустинг деревьев решений*

- Идея бустинга заключается в построении сильной предсказательной модели с использованием ансамбля более слабых
- Алгоритм использует *забывающие деревья решений* с небольшой глубиной в качестве слабых моделей

- **Забывающее дерево** — это упрощенная модель дерева решений, в которой каждый узел на одном уровне проверяет одно и то же условие ветвления
- Во время обучения деревья добавляются в ансамбль последовательно, и каждое дерево пытается исправить ошибки в предыдущем

Алгоритм CatBoost в двух словах

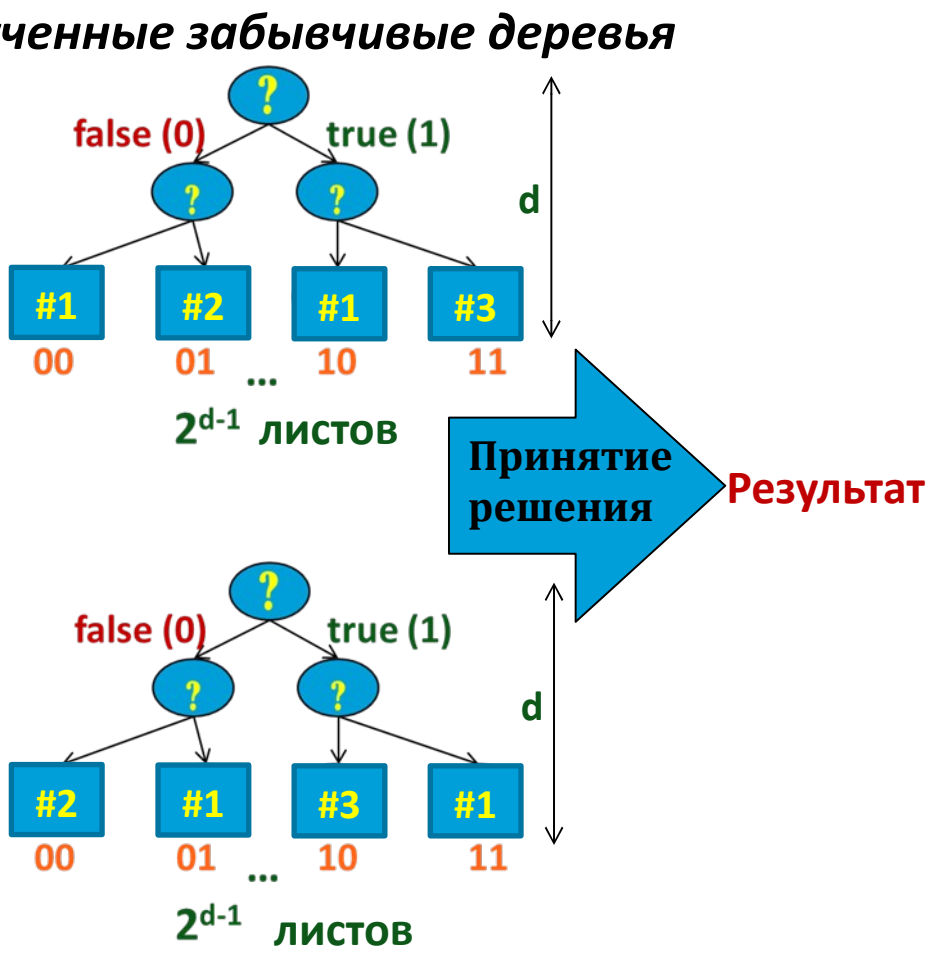
- Пример на задаче классификации:



Алгоритм CatBoost в двух словах

- Пример на задаче классификации:

  - Забывчивое дерево — это упрощенное дерево (одинаковые условия во всех узлах на одном уровне дерева)
  - Оптимизация (разработана Яндексом)
    - не хранить все деревья напрямую
    - исключить ветвление
    - достаточно отсортировать все предварительно обученные граничные значения и использовать побитовые операторы для поиска результирующего листа дерева
  - Наша цель: улучшить производительность на процессорах RISC-V без изменения алгоритма и структур данных («черный ящик»)



Тесты производительности

Datasets for performance analysis and optimization and their parameters. The maximum number of training iterations is set to 10000. Other parameters are set to default values.

Dataset	Rows x Cols	# of Classes	Loss Function	Learning rate	Tree depth
MQ2008	9630 x 46	-	YetiRank	0.02	6
Santander customer transaction	400k x 202	2	LogLoss	0.01	1
Covertype	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

- Мы рассматриваем несколько широко используемых наборов данных
- Мы не уверены, что эти наборы полностью репрезентативны, но, по крайней мере, мы охватываем *несколько сценариев моделирования*

Тесты производительности (2)

- «MQ2008»
  - 46 признаков для решения контролируемой задачи ранжирования
  - Содержит 9 630 обучающих и 2 874 тестовых образцов
- «Santander customer transaction»
  - 200 ненормализованных признаков вместе с двоичной целевой переменной
  - Обучающая и тестовая части имеют по 200 000 образцов
- «Covertype»
  - 52 целочисленных и бинарных признака, представляющих дикие местности и типы почв
  - необходимо для прогнозирования типа лесного покрова (7 классов)
  - Набор данных был случайным образом разделен на обучающий и тестовый наборы в соотношении 70:30

Тесты производительности (2)

- «YearPredictionMSD»
  - 90 ненормализованных признаков, извлеченных из песен
  - необходимо для прогнозирования года, в котором была выпущена песня
  - Набор данных разделен на 463 715 обучающих образцов и 51 630 тестовых образцов
- «image-embeddings»
  - подмножество набора данных PASCAL VOC 2007
  - содержит только изображения одного класса объектов из двадцати возможных
  - содержит 9 630 обучающих и 2 874 тестовых образцов
  - эмбеддинг был сгенерирован для изображений с использованием предварительно обученной модели resnet34 из библиотеки TorchVision
  - для получения эмбеддинга последний слой классификации был удален из модели

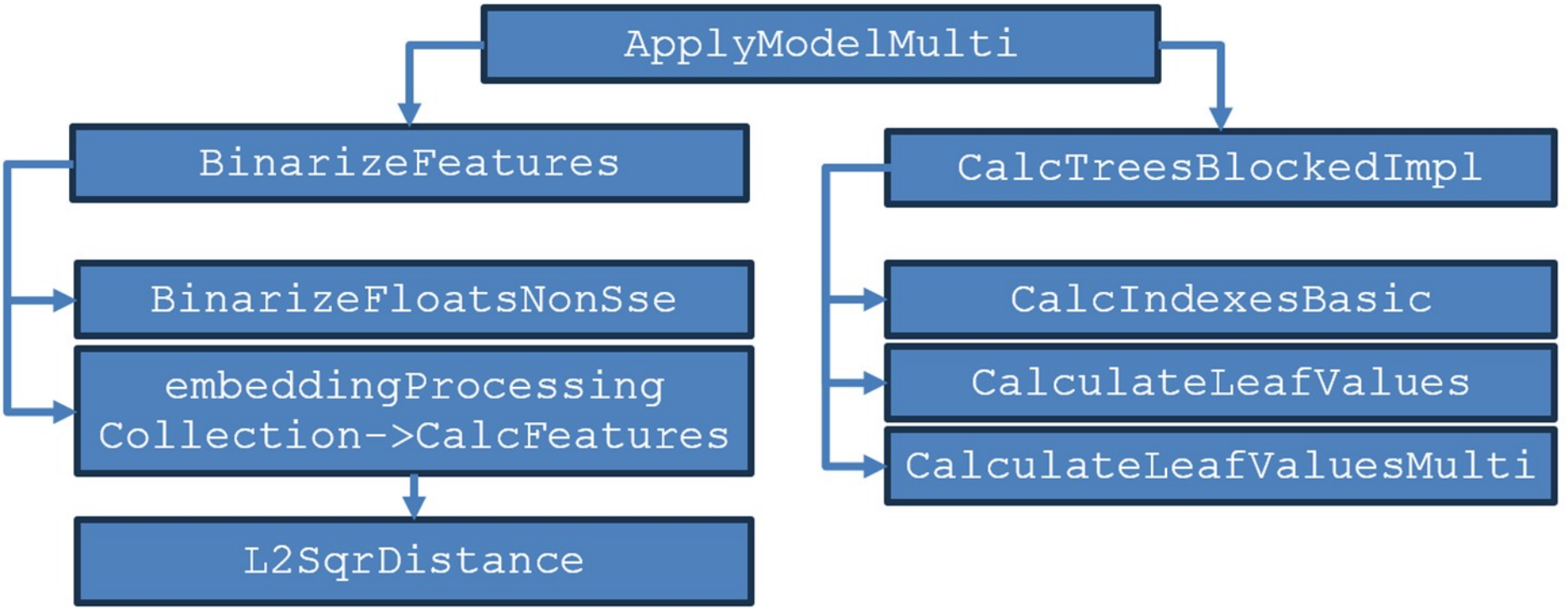
Возможности оптимизации

- **x86 CPU**: множество программных инструментов для профилирования и оптимизации (VTune, Advisor, C/C++ Compilers…)
- **RISC-V CPU**:
  - Компиляторы находятся на ранней стадии разработки (однако довольно неплохие)
  - Набор инструментов анализа производительности очень ограничен:
    - Функционал очень ограничен, можно найти только Hotspots
- Как оптимизировать большой код? *// Работает стандартный perf*
  - Найти Hotspots
  - Попробовать векторизацию циклов с интенсивными вычислениями
  - Проверить эффективность распараллеливания и улучшить при необходимости
- **Наша работа**: векторизация под **RVV 0.7.1**

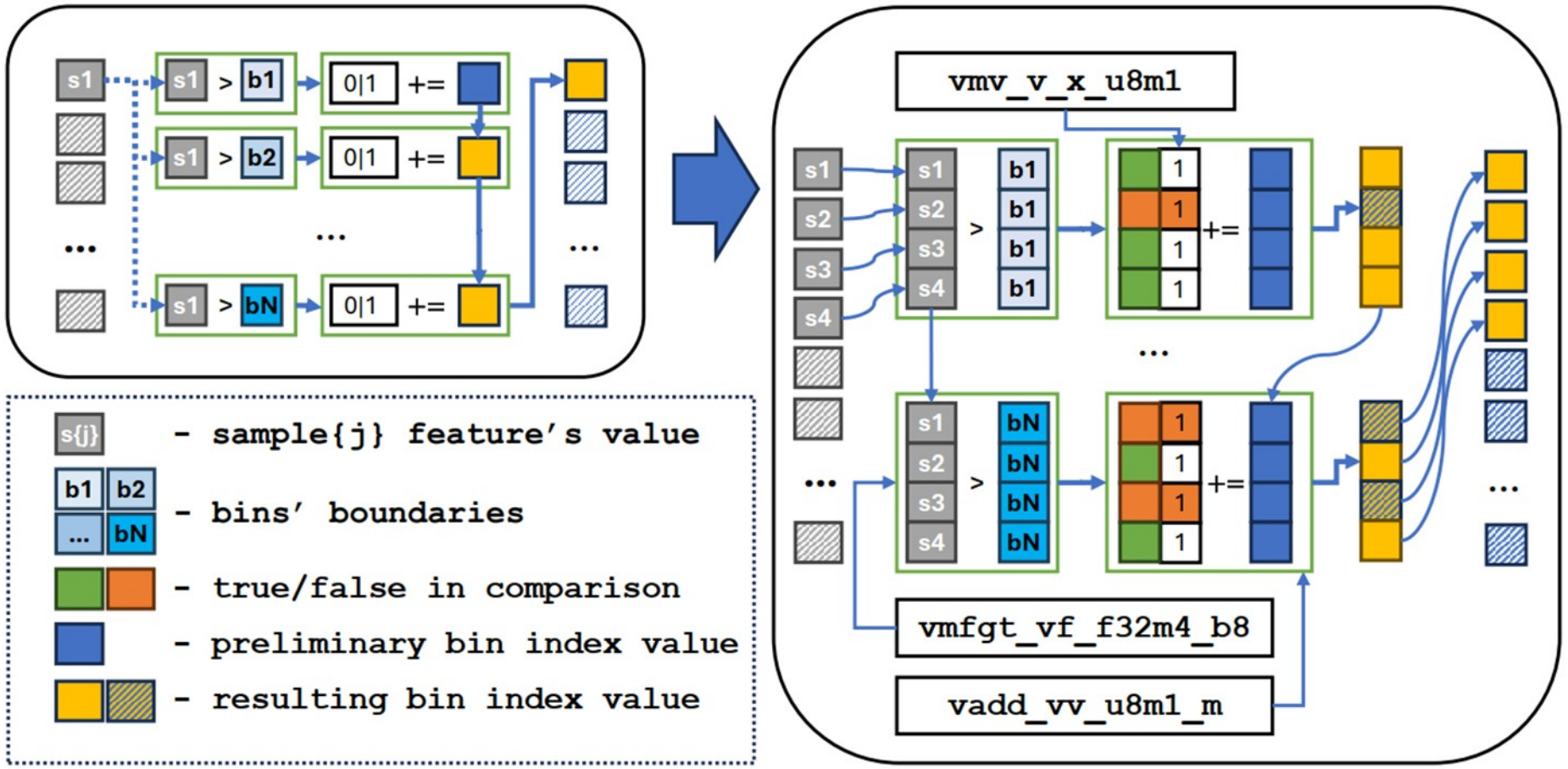
Как найти Hotspots?

- Использование инструмента perf из Linux для анализа CatBoost проблематично, так как он состоит из интерфейса Python и динамических библиотек на C++
- Технология профилирования:
  - Найти точку входа для анализа – наиболее трудоемкую функцию (perf)
  - Проанализировать тело функции, внедрить измерения времени и накопить данные в глобальные переменные (пользовательский таймер)
  - Проанализировать измерения кода/времени и построить граф вызовов
- Проверка:
  - Использовать те же инструменты на процессорах x86 и сравнивать с профилировщиками
  - Сравнивать запуски с профилированием и без него на процессорах RISC-V
  - Проведение финальных экспериментов без профилирования на полных наборах данных

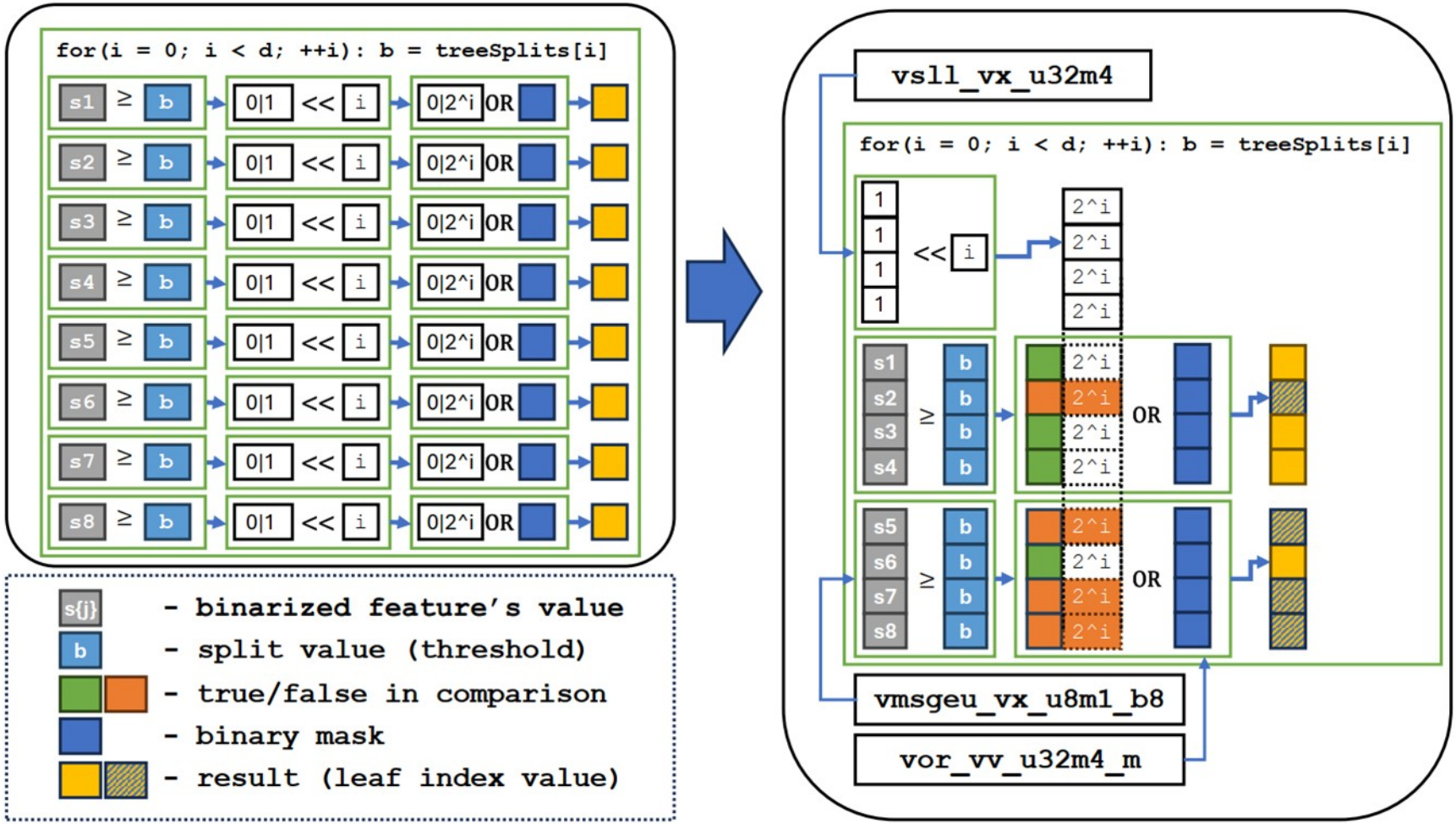
Кандидаты на улучшения производительности



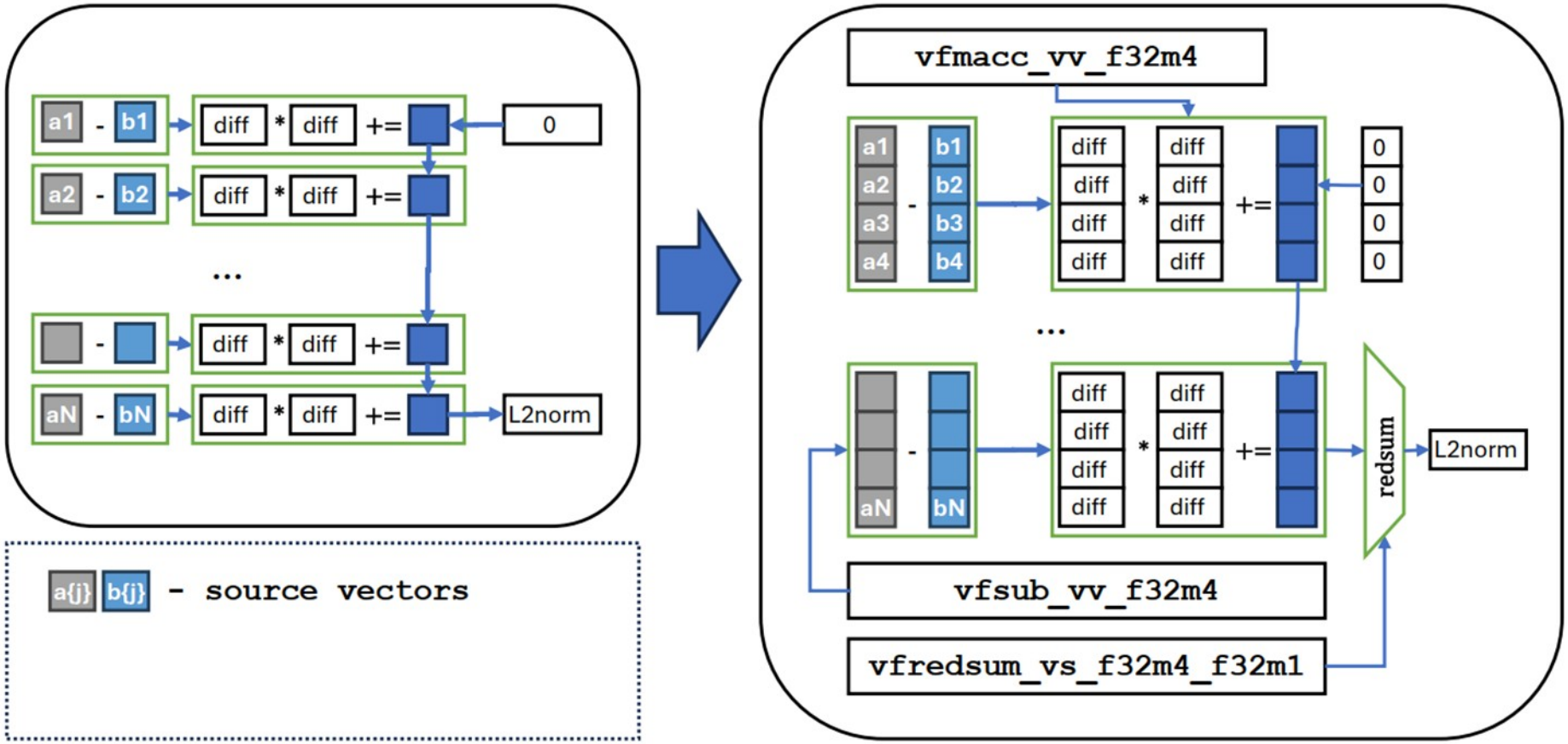
Векторизация BinarizeFloatsNonSse()



Векторизация CalcIndexesBasic()



Векторизация L2SqrDistance()



Векторизация для процессоров RISC-V

- Обратите внимание, что векторное расширение **RVV 0.7.1** позволяет *регулировать количество векторных регистров, используемых во время операций*.
- Например, мы могли бы использовать ***блок из четырех 128-битных регистров*** вместе с соответствующими блочными векторными операциями для обработки данных, как если бы архитектура содержала ***512-битные регистры***.
  - Этот метод соответствует суффиксу **m4** в именах встроенных функций.
- Выбор блока может *значительно* повысить производительность, но определение наилучшего режима (m1, m2, m4, m8) требует экспериментов.
- Мы используем вариант, который обеспечивает максимальную производительность.

Тестовая инфраструктура

- **x86** (обучение модели, подготовка наборов данных, вычислительная точность теста и сравнение производительности):
  - Intel Xeon Silver 4310T (2 ЦП по 10 ядер каждый, всего 20 ядер), 64 ГБ ОЗУ
- **RISC-V**: Мини-кластер *Lichee Cluster 4A*
  - 7 плат с ЦП RISC-V TH1520 на базе ядер C910
  - Каждый процессор содержит 4 ядра с поддержкой векторных расширений RVV 0.7.1
  - Каждая плата имеет 16 ГБ ОЗУ

Анализ производительности (1)

Profiling results of CatBoost prediction on the `YearPredictionMSD` dataset on RISC-V CPU. The code was run in a serial mode. Time is given in seconds.

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**

Анализ производительности (2)

Profiling results of CatBoost prediction on the `Covertype` dataset on RISC-V CPU. The code was run in a serial mode. Time is given in seconds.

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**

Анализ производительности (3)

Profiling results of CatBoost prediction on the `image-embedding` dataset on RISC-V CPU. The code was run in a serial mode. Time is given in seconds.

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 NonSse	312	0.03	0.13%	0.00	0.07%	**5.44**
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

Final comparison results. The code was run in a multithreaded mode. Time is given in seconds. An accuracy is same in all runs, therefore it is shown only once for each dataset.

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**
Covertype	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**

Производительность: заключение

- Время выполнения на сервере x86 указано только для справочных целей
- Результаты оптимизации кода показывают значительное (до 6 раз) ускорение для большинства наборов данных
- Два возможных ограничения использования наших результатов:
  - Ускорение достигается только в случае использования, когда модель работает с несколькими входами. Обычно при использовании одиночного входа выигрыш не ожидается
  - Hotspots и распределение вычислительной 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 процессоров RISC-V

Заключение

- Экосистема RISC-V развивается быстрыми темпами
- Текущий уровень развития инфраструктуры позволяет переносить программное обеспечение на существующие маломощные устройства RISC-V, а также выявлять наиболее перспективные подходы, характерные для RISC-V, для повышения производительности
- Мы обнаружили, что большой код можно сравнительно легко скомпилировать в RISC-V
- Из-за ограниченных возможностей компиляторов и инструментов производительности мы разработали собственную инфраструктуру профилирования и векторизовали код с использованием встроенных функций RVV 0.7.1, добившись ускорения до 5,8 раз
- Мы надеемся, что наш опыт будет полезен для переноса других фреймворков на процессоры с архитектурой RISC-V

Контакты

Нижегородский государственный университет
http://www.unn.ru

Институт информационных технологий, математики и механики
http://www.itmm.unn.ru

Мееров И.Б.
meerov@vmk.unn.ru