**Отзыв на статью**

**Системно-ориентированный параллельный алгоритм координатного спуска**

**(SySCD: A System-Aware Parallel**

**Coordinate Descent Algorithm)**

**Автор отзыва: Кошелев Сергей**

**Аннотация**

В статье была описана новая распараллеливающаяся версия алгоритма стохастического координатного спуска (СКС), которая обладает большей универсальностью и вероятностью сходимости. В начале статьи определяются слабые места СКС. Далее автор, основываясь на архитектуре процессора, предлагает несколько вариантов оптимизации алгоритма и в заключении приводит оценку полученных результатов на одном из датасетов Kaggle.

**Особенности алгоритма**

Современные вычислительные мощности предлагают широкие возможности ускорения программ благодаря распараллеливанию их работы. Одним из подобных алгоритмов является стохастический координатный спуск (СКС). Однако, подобный подход, несомненно, имеет ряд проблем, которые должны решиться благодаря применению системно-ориентированного СКС. Эти проблемы включают в себя: неэффективный доступ к кэш-памяти процессора, отсутствие универсальности применения многопоточности и неравномерный доступ к памяти.

Автор статьи привел примеры работы уже созданных асинхронных, синхронных и распределяемых алгоритмов, которые не могли обеспечить значительного ускорения обучения при увеличении числа потоков.

Чтобы найти способы оптимизации СКС, автор более подробно остановился на возможных слабых местах:

1. Когда вектор модели не помещается в кэше процессора (строка процессора очень маленькая), тратится много времени на доступ к модели. Эта проблема характерна как для параллельной, так и для последовательной реализаций.
2. При параллельной реализации есть проблема с доступом к общему вектору **v**, что может привести даже к расходимости.
3. Неравномерный доступ к памяти, т.к. задержка между узлами сильно больше, чем внутри них.
4. Перестановка координат занимает слишком много времени.

Для их решения автор привёл несколько методов оптимизации:

1. Сегментированная оптимизация  
   Для смягчения проблем №1 и №4 предлагается разделить координаты и соответствующие столбцы на сегменты и далее обучить весь сегмент отдельно за один раз. Благодаря этому кэш-линии хватит для вектора модели, уменьшатся издержки рандомизации координат, увеличится предварительная эффективность ЦПУ.
2. Увеличение параллельности данных  
   Эта оптимизация направлена на смягчение проблемы №2. В ней предлагается избавиться от конфликта записи в общий вектор **v**, копируя этот вектор между потоками. Общий вектор доступен только во время упрощенных интервалов времени, таких как границы эпохи. Для улучшения сходимости по сравнению с распределяемыми алгоритмами предлагается перетасовывать сегменты в начале каждого прохода, а затем каждому потоку выбирать новый сегмент.
3. Оптимизация неравномерного доступа к памяти  
   Чтобы избежать проблемы №3 добавляется еще один уровень распараллеливания.

Для оценки качества алгоритма было проведено сравнение с другими известными реализациями на 2 различных архитектурах и на 3 различных наборах данных. По проведенным тестам подтверждено, что системно-адаптированный СКС линейно масштабируется между системами или ядрами, т.е. успешно решена проблема №2. Во-вторых, эта реализация была сравнена автором с обычной и показала ту же правильность решения. Наконец, алгоритм был сравнен с популярными фреймворками. В среднем, при тех же начальных условиях, модифицированный СКС оказался в 10 раз быстрее.

**Заключение**

Автором было показано, что существующие алгоритмы параллельного координатного спуска сильно страдают от своих слабых мест, связанных с архитектурой процессоров. Это не позволяло полностью реализовать потенциал этого алгоритма. Предложенное решение всех перечисленных проблем помогло ускорить работу примерно в 12 раз, а также увеличить вероятность сходимости и универсальность алгоритма.