УДК 004.75

Глотов Е.С.[1], Герасимов С.В.[1], Мещеряков А.В.[2,3]

[1] Московский государственный университет им. М.В. Ломоносова, г. Москва, Россия
[2] Институт космических исследований Российской академии наук, г. Москва, Россия
[3] Казанский (Приволжский) федеральный университет, Казань, Россия

РАСПРЕДЕЛЕННЫЙ МЕТОД СОПОСТАВЛЕНИЯ АСТРОНОМИЧЕСКИХ КАТАЛОГОВ НА ПЛАТФОРМЕ APACHE SPARK*

АННОТАЦИЯ

В работе предложен горизонтально-масштабируемый алгоритм сопоставления астрономических каталогов, реализованный на платформе распределенных вычислений Apache Spark. Метод обеспечивает необходимую точность сопоставления каталогов и хорошую производительность в сравнении с лучшими реализациями подобных систем, доступными в астрономическом сообществе. Горизонтальная масштабируемость предложенного метода была подтверждена с помощью экспериментов на кластере, развёрнутом в облаке Microsoft Azure.

КЛЮЧЕВЫЕ СЛОВА

Астрономические каталоги; распределённые вычисления; облачные вычисления; Apache Spark; HEALPix.

Evgeniy Glotov[1], Sergey Gerasimov[1], Alexander Meshcheryakov[2,3]

[1] Lomonosov Moscow State University, Moscow, Russia
[2] Space Research Institute of the Russian Academy of Sciences, Moscow, Russia
[3] Space Kazan (Volga Region) Federal University, Kazan, Russia

DISTRIBUTED METHOD FOR CROSSMATCHING OF ASTRONOMICAL CATALOGS BASED ON APACHE SPARK PLATFORM

ABSTRACT

In the article a horizontally-scalable algorithm for matching astronomical catalogs, based on Apache Spark, distributed computing platform, is proposed. The method provides the necessary accuracy and good performance compared to the best systems available in the astronomical community. Horizontal scalability of the proposed method was confirmed by experiments on a cluster deployed in the Microsoft Azure cloud service.

KEYWORDS

Cross-matching; astronomical catalogs; distributed computing; cloud computing; Apache Spark; HEALPix.

Введение

Основным источником данных о небесных объектах в современной наблюдательной астрофизике являются цифровые обзоры неба, представляющие собой наборы большого числа астрономических изображений, полученных телескопом в заданном спектральном диапазоне, и покрывающие значительные участки звездного неба. Для поиска астрономических объектов и извлечения их признаков из изображений астрономы используют последовательность преобразований - т.н. конвейер обработки астрономических изображений (подробнее см., например, [1]). Конечная цель работы конвейера – получение каталога астрономических объектов.

Астрономический каталог представляет собой таблицу всех обнаруженных на изображениях неба объектов (галактик, звезд, астероидов и др.); множество столбцов таблицы представляет собой фиксированный набор свойств, выделенных из изображения объекта: небесные координаты, яркость, размер, форма, морфология, и другие признаки. В настоящее время

* **Труды I Международной научной конференции «Конвергентные когнитивно-информационные технологии» (Convergent'2016), Москва, 25-26 ноября, 2016**

в общедоступных астрономических архивах (см., например, [2]) собраны более 10 тыс. каталогов, полученных на основе данных наблюдений из небесных обзоров. Необходимо отметить, что объем данных небесных обзоров и их каталогов сейчас переживает взрывной рост. Так, самый известный оптический обзор неба на сегодняшний день, SDSS-III [3], на момент своего завершения в 2013 году содержал астрономический каталог размером 3.4 ТБ. В ключевом проекте следующего десятилетия, в обзоре неба телескопом LSST [4] ожидаемый объём каталогов составит около 15 петабайт и будет содержать информацию о более чем 350 миллиардах объектов.

Увеличение количества независимых источников информации об известных астрономических объектах (данные наблюдений в разных спектральных диапазонах, полученные в разное время, разными телескопами) критически важно для решения задачи их точной классификации и измерения физических характеристик. Также важной практической задачей для современных небесных обзоров является поиск новых астрономических объектов в непрерывном потоке данных наблюдений, поступающих в виде астрономических каталогов, непосредственно с телескопа. Для поиска новых небесных объектов новый каталог, поступивший с телескопа, необходимо сопоставить в реальном времени с астрономическими каталогами, полученными ранее. Таким образом, задача сопоставления астрономических каталогов имеет важное практическое значение для наблюдательной астрофизики.

Современные объемы данных астрономических каталогов из небесных обзоров требуют распределенного хранения. В свете экспоненциального роста ожидаемого объема данных астрономических каталогов в ближайшем будущем, при построении системы для сопоставления астрономических каталогов необходимо заложить требование ее горизонтальной масштабируемости.

Целью настоящей работы является разработка распределённого горизонтально-масштабируемого метода сопоставления астрономических каталогов. Предложенный здесь метод сопоставления каталогов разрабатывался для встраивания в распределённую систему обработки и анализа больших массивов астрономических данных [1] на платформе Apache Spark [5].

Статья построена следующим образом. В следующем разделе дана формальная постановка задачи сопоставления астрономических каталогов и сделан обзор имеющихся подходов к ее решению. Далее описан предлагаемый нами распределенный метод сопоставления астрономических каталогов на платформе Apache Spark. Затем представлены выполненные эксперименты, которые подтверждают точность и горизонтальную масштабируемость предложенного метода. В последнем разделе представлены наши выводы.

Задача сопоставления астрономических каталогов и методы ее решения

В настоящей работе будет рассматриваться классическая задача сопоставления (кросс-отождествления) двух астрономических каталогов по небесным координатам, которая может быть формализована следующим образом.

Определения:

- Астрономический объект – кортеж, состоящий из идентификатора, сферических координат RA (прямое восхождение) и DEC (склонение), и прочих характеристик небесного объекта: X= [$id, ra, dec, [features]$];
- Каталог астрономических объектов – двумерная таблица, строками которой являются астрономические объекты.

Дано:

- A, B – каталоги астрономических объектов;
- $dist(X,Y) = haversine(X,Y)$ - функция углового расстояния на сфере между астрономическими объектами X и Y;
- $radius$ – максимально допустимое угловое расстояние между отождествлёнными объектами (радиус сопоставления).

Найти каталог астрономических объектов C:

$$C = \{ [X, Y]: X \in A, Y \in B, \forall W \in B: dist(X, Y) < dist(X, W), \forall U \in A: dist(X, Y) < dist(U, X), dist(X, Y) < radius\}.$$

Качество сопоставления оценивается по следующим метрикам:

- Точность (precision) = $\frac{Count_{real\ \&\ found}}{Count_{found}}$;
- Полнота (recall) = $\frac{Count_{real\ \&\ found}}{Count_{real}}$,

где числа $Count_{real}$, $Count_{found}$ и $Count_{real\ \&\ found}$ представляют собой, соответственно: точное число сопоставлений (пар объектов из двух каталогов), найденное число сопоставлений

предложенным нами методом и найденное число сопоставлений одновременно являющимися точными. Для поиска точных сопоставлений двух каталогов, в качестве эталона, мы будем использовать популярную программу TOPCAT [6] (см. раздел Эксперименты). В рамках формальной постановки задачи, представленной выше, мы ввели важное требование для разрабатываемого метода: точность и полнота сопоставления должны составлять 100%.

Отметим, что в такой постановке задача сопоставления двух астрономических каталогов сводится к задаче поиска для каждого объекта первого каталога ближайшего соседа внутри радиуса сопоставления среди всех объектов второго каталога. Ниже рассмотрены несколько распространенных методов сопоставления астрономических каталогов, которые находят применение в современных астрономических исследованиях.

В статье [7] представлен алгоритм сопоставления каталогов с помощью скользящего окна. Первым этапом алгоритма является сортировка объектов в каталогах по значению их склонения. Далее для каждого окна выбирается минимальное и максимальное значение склонения, находятся минимальный и максимальный порядковые номера объектов обоих каталогов, значение склонения которых входит в окно. Для каждого объекта первого каталога выбираются близкие по координате RA объекты второго каталога, среди них при помощи перебора находится ближайший. Вычислительная сложность данного алгоритма составляет $O(n \log m)$, где m, n – число объектов 1 и 2 каталога. К недостаткам метода можно отнести сложность распараллеливания (в виду сильного пересечения по данным между соседними окнами), что затрудняет создание его эффективной распределённой реализации.

Современные реляционные СУБД позволяют строить пространственный индекс на разнообразных древовидных иерархических структурах данных (например, R-деревьях - см. [8]). В частности, сопоставление каталогов в популярной астрономической СУБД CasJobs [9] может осуществляться с использованием быстрого поиска ближайшего соседа на основе пространственного индекса.

Эффективным решением для построения распределенных алгоритмов сопоставления каталогов является разбиение небесной сферы на неперекрывающиеся области и сопоставление объектов в каждой области независимо от других. При таком подходе необходимо предусмотреть дополнительную обработку объектов из разных областей, расположенных рядом с границей (далее – граничные эффекты). В статье [10], небесная сфера разбивалась на срезы заданной ширины по одной из небесных координат (DEC). В работе [11] было предложено иерархическое разбиение небесной сферы HEALPIX (англ. Hierarchical Equal Area isoLatitude PIXelization), каждый уровень которого разбивает сферу на клетки (ячейки разбиения) равной площади (рис. 1). Использование HEALPix позволяет снизить затраты на обработку граничных эффектов (см., например, [12]).

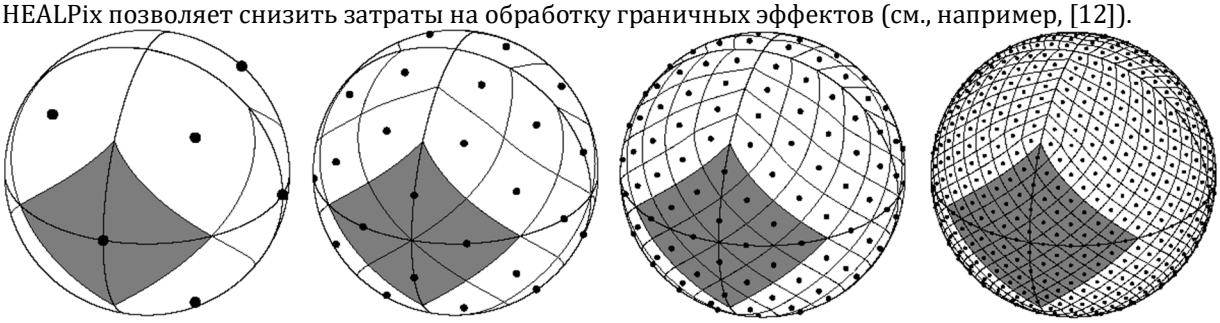

Рис. 1. Первые 4 уровня разбиения HEALPix

В работах [12] и [13] предложены схожие алгоритмы сопоставления астрономических каталогов на основе HEALPix, состоящие из следующих этапов: (i) считывание координат объектов и получение индексов HEALPix для каждого объекта, (ii) группировка объектов по ключу - индексу HEALPix, (iii) поиск сопоставления между объектами двух каталогов внутри каждой клетки HEALPix с учетом объектов на границах соседних клеток, (iv) объединение результатов. Данный подход широко используется при сопоставления астрономических каталогов. В частности, он реализован в наиболее популярных приложениях для анализа астрономических каталогов: TOPCAT [6], CDS X-Match [14], Large Survey Database [15].

Предложенное решение

Система обработки астрономических данных [1] построена на базе фреймворка распределённых вычислений Apache Spark [5].

Apache Spark предоставляет широкий набор возможностей для распределённой обработки данных, включая возможность чтения и записи в распространённые распределённые файловые системы, кэширование данных в памяти для многоразового использования, удобный механизм

использования сторонних библиотек, возможность работы в интерактивном режиме, а также большой набор встроенных методов обработки данных, включая методы интеллектуального анализа данных. Spark – это масштабируемая платформа анализа данных, которая включает в себя примитивы для вычислений в оперативной памяти. Основной абстракцией Spark являются устойчивые распределенные наборы данных (RDD, англ. Resilient Distributed Datasets).

При разработке распределённого алгоритма сопоставления каталогов астрономических объектов на базе Apache Spark за основу был взят метод, предложенный в статье [13]. Указанный подход позволяет разбить задачу на большое число подзадач, каждая из которых выполняется практически независимо от других. Способ обработки граничных эффектов, выбранный в данном решении, также представляется эффективным и имеет низкие накладные расходы (см. ниже в разделе Эксперименты).

Предлагаемый нами алгоритм сопоставления каталогов состоит из следующих этапов:

1. Считывание объектов из каталогов в распределённые массивы (RDD);
2. Нахождение для каждого объекта индексов HEALPix (клеток) с учётом граничных эффектов и возможной фильтрацией;
3. Группировка объектов в новые массивы по ключу – индексу клетки;
4. Соединение по ключу двух сгруппированных массивов;
5. Поиск наилучшего сопоставления между объектами каталогов в пределах каждой клетки;
6. Устранение дубликатов, появившихся при учёте граничных эффектов;
7. Запись отобранных пар объектов в объединённый каталог.

В качестве языка реализации алгоритма в Apache Spark был выбран язык программирования Java. Ниже подробно рассмотрены этапы, непосредственно реализующие логику метода.

Учёт граничных эффектов. Предложенный метод позволяет учесть граничные эффекты с минимальными накладными расходами. Учёт границ проводится только для того каталога, для которого ведётся поиск сопоставлений в другом каталоге. Границей для клетки HEALPix некоторого уровня разбиения назовем все клетки HEALPix более высокого уровня разбиения, окрестность которых не полностью входит внутрь данной клетки (рис. 2). Каждый объект нужно поместить во все клетки низкого уровня разбиения, с которыми пересекается окрестность его клетки высокого уровня разбиения. Уровень разбиения для границы выбирается так, чтобы клетка HEALPix данного уровня вмещала не менее четырех площадей кругов сопоставления объектов ($4\pi * radius^2$) в каталогах. Тогда можно гарантировать, что для всех объектов будет найдено глобально наилучшее сопоставление, а не локально лучшее внутри конкретной клетки. Для выбранного нами радиуса сопоставления каталогов, равного 1 секунде дуги, для учета граничных эффектов был выбран 16 уровень разбиения HEALPix. Следует отметить, что число дубликатов, появившихся в результате учёта граничных эффектов, экспоненциально зависит от разницы между базовым и граничным уровнями разбиения. Например, если при разнице в 12 уровней разбиения число дубликатов составляет всего 0.1% от общего числа объектов, то при разнице в 4 уровня дублирование составляет уже 26%, что является значительным показателем.

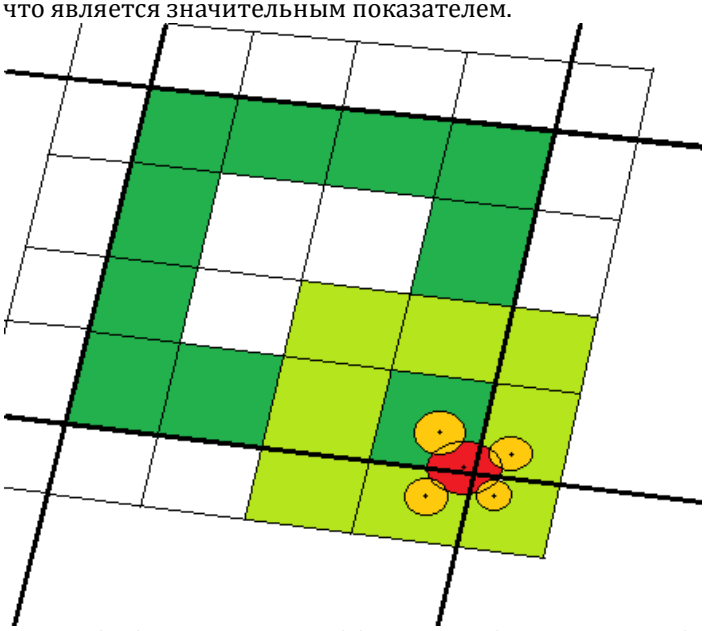

Рис. 2. Обработка граничных эффектов в предложенном методе

Фильтрация. Эта операция может быть полезна в случае если первый астрономический каталог существенно меньше второго, либо сосредоточен в определенной области неба. В этом случае, множество клеток HEALPix, в которых расположены объекты первого каталога, невелико. Фильтрация заключается в отсеивании объектов из второго каталога, расположенные в клетках, не граничащих с клетками для объектов первого каталога. Фильтрация позволяет иногда значительно ускорить дальнейшие шаги алгоритма, начиная с группировки объектов по клеткам разбиения (сокращение размера второго каталога позволит быстрее произвести сопоставление).

Поиск наилучшего сопоставления внутри клетки. После того, как объекты сгруппированы в массивы по клеткам (с учётом граничных эффектов), каждую клетку можно обрабатывать независимо от других. Чтобы сопоставить каталоги «A» и «B» в клетке разбиения с индексом «i», нужно найти для каждого объекта из «A_i» ближайший к нему объект из «B_i». Алгоритм состоит из следующих шагов:

1. Сортировка каталога «A_i» по склонению (dec)
2. $\forall\, X_B \in$ «B_i»: бинпоиск $X_A \in$ «A_i»: $|X_A.\ dec - X_B.dec| < radius$;
3. $\forall\, X_B \in$ «B_i»: $\forall\, X_A \in$ «A_i»: $|X_A.\ dec - X_B.dec| < radius$:
   Если $dist(X_A, X_B) < minDist(X_A, B)$, то обновляем ближайший к X_A объект;
4. На выходе массив X_B, ближайших к каждому X_A.

Сложность алгоритма сопоставления – $O((m+n)\log m)$, где m, n – число объектов 1 и 2 каталога, расположенных в клетке с индексом «i».

Устранение дубликатов. Дубликаты появляются в результате учёта граничных эффектов. Для каждого объекта первого каталога может возникнуть от 1 до 4 сопоставлений из второго каталога. Для устранения дубликатов требуется сгруппировать массив пар объектов по ключу – идентификатору первого объекта из пары, выбрать объект, ближайший к искомому объекту в его группе, и записать его в результирующий массив.

Эксперименты

Для экспериментов по сопоставлению каталогов астрономических объектов были выбраны следующие каталоги:

- GALEX DR6: 222 554 652 объекта, общий размер файлов - 11.3 ГБ
- SDSS DR12: 794 013 968 объектов, общий размер файлов - 40 ГБ
- 3XMM DR5: 396 910 объектов, общий размер файлов - 18.7 МБ

Также для проведения экспериментов по определению зависимости скорости работы метода от данных из каталогов GALEX и SDSS были случайным образом отобраны половина и четверть объектов, и из них были созданы дополнительные каталоги.

Сравнительно небольшой объём данных обусловлен тем фактом, что для реализации алгоритма сопоставления каталогов в классической постановке задачи объект должен содержать всего 3 поля: идентификатор (ID), значение прямого восхождения (right ascension, RA) и склонения (declination, DEC). Идентификатор кодируется 64-разрядным целым беззнаковым числом, а координаты – числом с плавающей точкой двойной точности. Остальные признаки могут использоваться для применения новых техник сопоставления с целью улучшения качества, но их рассмотрение выходит за рамки данной статьи.

Для сравнения качества и скорости работы предложенного метода с существующими решениями, работающими на локальной машине, развёрнут тестовый стенд со следующими характеристиками:

Аппаратура:

- Intel core i7 4770k (4 ядра, 4 ГГц);
- 32ГБ ОЗУ (1866 МГц);
- SSD Kingston KC400 (550/530 МБ/с (чтение/запись)).

Программное обеспечение:

- Windows 10;
- TOPCAT;
- Java Runtime Environment 7;
- Apache Spark 2.0.0: локальный режим, 1 – 4 ядра, 5ГБ ОЗУ на ядро.

Было произведено сравнение результата выполнения предложенного алгоритма с результатами выполнения программы TOPCAT в качестве эталонного решения. В результате сопоставления каталогов 3XMM и ¼ SDSS в обоих случаях выявлено 14492 сопоставления, все сопоставления, обнаруженные TOPCAT, были обнаружены предложенным алгоритмом. В результате сопоставления каталогов ¼ GALEX и ¼ SDSS обе программы выдали 2558814 сопоставления, все обнаруженные сопоставления совпали. Результаты данных экспериментов

подтверждают качество обработки граничных эффектов предложенным методом, благодаря чему полнота и точность метода составляют 100% в рамках формальной постановки задачи.

На рис. 3 представлено сравнение скорости работы TOPCAT и предложенного метода на локальной машине. Предложенный метод на одном ядре процессора работает быстрее в 2.8 раза за счёт эффективного соединения каталогов и поиска ближайших соседей. При выделении программе всех четырёх ядер процессора скорость работы возросла ещё в 3.7 раз.

Так как алгоритм использует разбиение HEALPix, возникла необходимость в оценке скорости работы в зависимости от используемого уровня разбиения. Число клеток HEALPix находится в степенной зависимости от уровня разбиения, а число объектов в одной клетке обратно пропорционально числу клеток. Число клеток напрямую влияет на степень параллелизма алгоритма, а число объектов в клетке влияет на время обработки каждого из параллельных вычислителей. Также следует отметить, что из-за неравномерности покрытия неба цифровыми обзорами, в некоторых клетках может содержаться значительно больше объектов, чем в остальных. Таким образом, критический путь алгоритма зависит от баланса между размером клетки и числом объектов в ней. На третьем уровне разбиение HEALPix содержит 768 клеток, рассматривать уровни ниже не имеет смысла из-за недостаточной степени параллелизма. На 12 уровне разбиение содержит более 200 миллионов клеток, что практически равносильно числу объектов в исходных каталогах, и уровни выше рассматривать не имеет смысла, так как из-за них будет происходить повышение объёма данных, а не его понижение. Наилучшую скорость алгоритм показал при использовании HEALPix уровня 5 (рис. 4). Число клеток на данном уровне составило 12288, что является хорошим ресурсом параллелизма. Время сопоставления каталогов SDSS и GALEX составило 673 секунды. Число дублей при учёте граничных эффектов составило 0.4%.

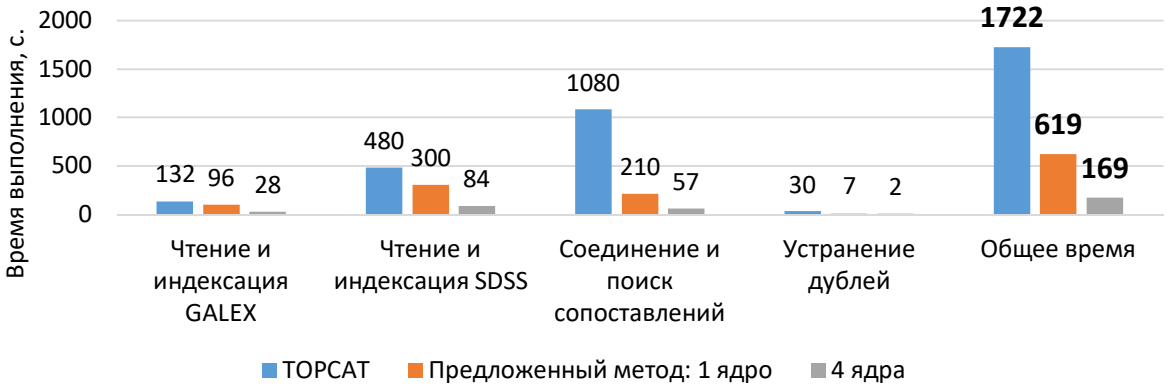

Рис. 3. Сравнение времени сопоставления ¼ каталогов GALEX и SDSS в TOPCAT и предложенным методом на 1 и 4 ядрах процессора

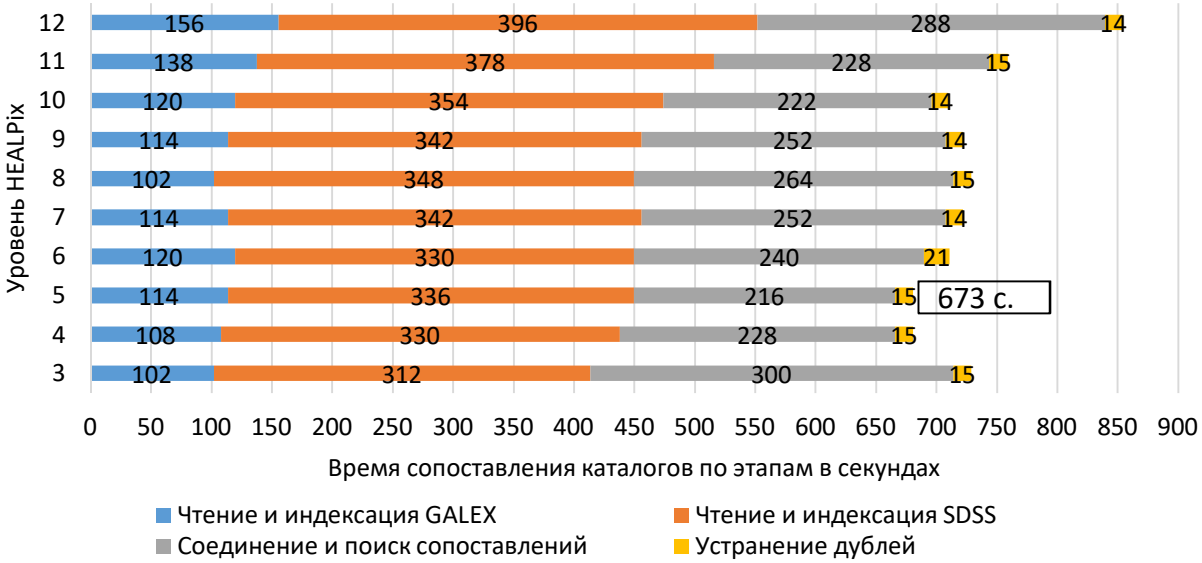

Рис. 4. Время сопоставления каталогов GALEX DR6 и SDSS DR12 в зависимости от уровня HEALPix

На рис. 5 представлена зависимость скорости сопоставления каталогов предложенным

методом в зависимости от числа объектов. Наблюдается линейная зависимость скорости работы метода от объёма входных данных.

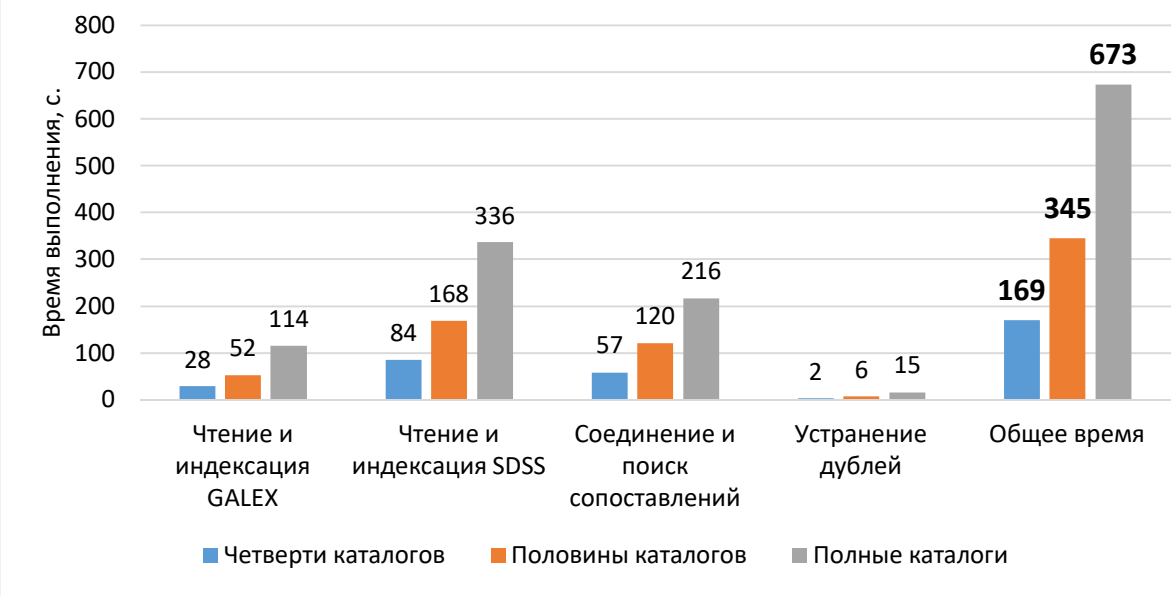

Рис. 5. Скорость сопоставления каталогов предложенным методом в зависимости от числа объектов

Эксперимент по оценке влияния фильтрации на скорость работы алгоритма проводился на каталогах 3XMM и SDSS. Сопоставление каталога небольшого размера с каталогом большого размера с использованием фильтрации дало прирост скорости в 2.8 раза. При этом считывание большого каталога ускорилось в 2 раза, а само сопоставление составило 2 секунды вместо 2 минут (рис. 6).

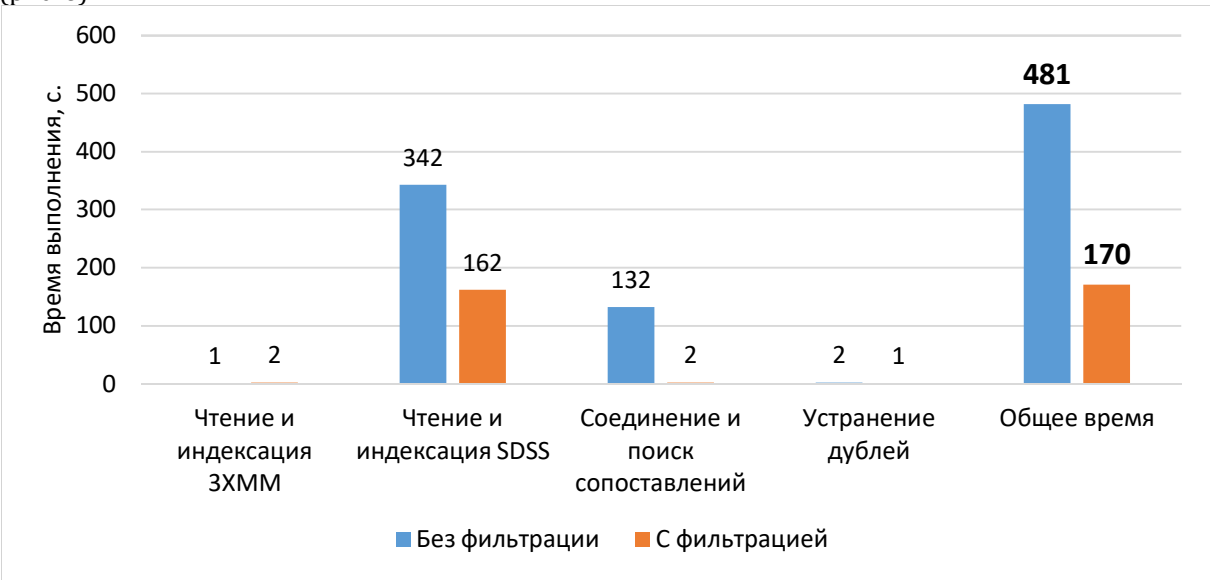

Рис. 6. Сравнение времени сопоставления каталогов 3XMM и SDSS с фильтрацией и без

Для оценки масштабируемости метода в инфраструктуре облачных вычислений Microsoft Azure был развёрнут тестовый стенд. Его аппаратная составляющая представляет собой кластер из связанных локальной сетью узлов в количестве от 2 до 12 со следующими характеристиками:

- 1/3 Intel Xeon E5-2673 v3 (4 ядра, 2.4 ГГц);
- 28ГБ ОЗУ;
- SSD.

Программное обеспечение:

- Java Runtime Environment 7;
- Apache Spark 2.0.0: От 8 до 48 потоков, по 4 потока на узел, 5ГБ ОЗУ на ядро.

Входные и выходные данные размещаются в облачном хранилище данных, связанном с каждым узлом кластера. Скорость чтения и записи в хранилище составляет 100МБ/с на любом узле кластера.

Эксперименты по оценке масштабируемости алгоритма проводились на полных каталогах

GALEX и SDSS. На одном и том же числе потоков кластер показал скорость работы около 2 раз ниже, чем на локальной машине. Это связано с тем, что хранилище кластера имеет скорость доступа в несколько раз ниже, чем накопитель, установленный на локальной машине, процессоры узлов кластера имеют более низкую тактовую частоту, а также на локальной машине нет необходимости передавать данные по сети. Также результаты экспериментов (рис. 7) показывают линейную зависимость скорости работы алгоритма от числа узлов в кластере.

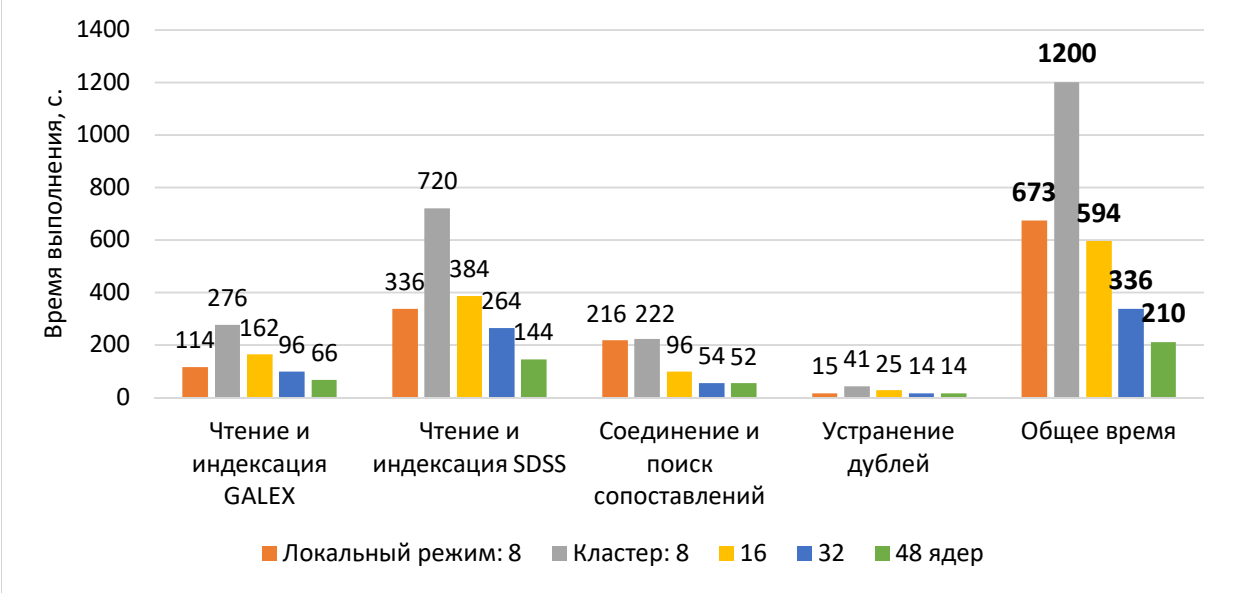

Рис. 7. Скорость сопоставления каталогов предложенным методом в зависимости от числа ядер в кластере

Выводы

В работе предложен горизонтально-масштабируемый алгоритм сопоставления астрономических каталогов, реализованный на платформе распределенных вычислений Apache Spark. Метод обеспечивает необходимую точность сопоставления каталогов и хорошую производительность в сравнении с лучшими реализациями подобных систем, доступными в астрономическом сообществе. В частности, в нераспределенном режиме созданный алгоритм в несколько раз превосходит по скорости сопоставления каталогов популярную систему TOPCAT. Горизонтальная масштабируемость предложенного метода была подтверждена с помощью экспериментов на кластере, развёрнутом в облаке Microsoft Azure.

Мы видим следующие направления развития предложенного здесь базового метода распределенного сопоставления астрономических каталогов:

- Исследование возможности дальнейшего улучшения предложенного алгоритма в части скорости сопоставления объектов внутри клеток разбиения (см., например, [13]);
- Исследование проблемы сопоставления более двух каталогов (см., например, [16]) в реальном времени;
- Использование при сопоставлении каталогов дополнительных свойств объектов (помимо небесных координат), что позволит снять возможную проблему неоднозначности кросс-отождествления объектов исключительно по координатам.

Работа поддержана Российским фондом фундаментальных исследований (гранты РФФИ №14-22-03111 офи_м и №15-29-07085 офи_м). Авторы благодарят компанию Microsoft за облачные ресурсы, предоставленные коллективу в рамках программы "Microsoft Azure for Research". А.В. Мещеряков также благодарен финансированию за счет средств субсидии в рамках государственной программы повышения конкурентоспособности Казанского (Приволжского) федерального университета.

Литература

1. Герасимов С.В. и др. Архитектура системы обработки больших массивов астрономических данных //Материалы 4-й Всероссийской научно-технической конференции «Суперкомпьютерные технологии» (СКТ-2016). Ростов-на-Дону, 2016. Т. 2. С. 144-148.
2. Strasbourg astronomical Data Center (CDS) // URL:http://vizier.u-strasbg.fr/viz-bin/VizieR.

3. Alam S. et al. The eleventh and twelfth data releases of the Sloan Digital Sky Survey: Final data from SDSS-III //The Astrophysical Journal Supplement Series. – 2015. – Т. 219. – №. 1. – С. 12.
4. Ivezic Z. et al. Large Synoptic Survey Telescope: From science drivers to reference design //Serbian Astronomical Journal. – 2008. – Т. 176. – С. 1-13.
5. Zaharia M. et al. Spark: cluster computing with working sets //HotCloud. – 2010. – Т. 10. – С. 10-10.
6. Taylor M. TOPCAT: tool for operations on catalogues and tables //Astrophysics Source Code Library. – 2011. – Т. 1. – С. 01010.
7. Devereux D. et al. An O (N log M) Algorithm for Catalogue Crossmatching //Astronomical Data Analysis Software and Systems XIV. – 2005. – Т. 347. – С. 346.
8. Guttman A. R-trees: a dynamic index structure for spatial searching. – ACM, 1984. – Т. 14. – №. 2. – С. 47-57.
9. Li N., Szalay A. CASJobs: A workflow environment designed for large scientific catalogs //2008 Third Workshop on Workflows in Support of Large-Scale Science. – IEEE, 2008. – С. 1-8.
10. Nieto-Santisteban M. A., Thakar A. R., Szalay A. S. Cross-matching very large datasets //National Science and Technology Council (NSTC) NASA Conference. – 2007.
11. Gorski K. M. et al. HEALPix: a framework for high-resolution discretization and fast analysis of data distributed on the sphere //The Astrophysical Journal. – 2005. – Т. 622. – №. 2. – С. 759.
12. Zhao Q. et al. A paralleled large-scale astronomical cross-matching function //International Conference on Algorithms and Architectures for Parallel Processing. – Springer Berlin Heidelberg, 2009. – С. 604-614.
13. Pineau F. X., Boch T., Derriere S. Efficient and Scalable Cross-Matching of (Very) Large Catalogs //Astronomical Data Analysis Software and Systems XX. – 2011. – Т. 442. – С. 85.
14. CDS X-Match service // URL:http://cdsxmatch.u-strasbg.fr/xmatch.
15. Juric M. Large Survey Database // URL:http://research.majuric.org/trac/wiki/LargeSurveyDatabase.
16. Pineau F. X. et al. Probabilistic multi-catalogue positional cross-match //arXiv preprint arXiv:1609.00818. – 2016.

References

1. Gerasimov S.V. i dr. Arkhitektura sistemy obrabotki bol'shikh massivov astronomicheskikh dannykh //Materialy 4-y Vserossiyskoy nauchno-tekhnicheskoy konferentsii «Superkomp'yuternye tekhnologii» (SKT-2016). Rostov-na-Donu, 2016. T. 2. S. 144-148.
2. Strasbourg astronomical Data Center (CDS) // URL:http://vizier.u-strasbg.fr/viz-bin/VizieR.
3. Alam S. et al. The eleventh and twelfth data releases of the Sloan Digital Sky Survey: Final data from SDSS-III //The Astrophysical Journal Supplement Series. – 2015. – T. 219. – №. 1. – S. 12.
4. Ivezic Z. et al. Large Synoptic Survey Telescope: From science drivers to reference design //Serbian Astronomical Journal. – 2008. – T. 176. – S. 1-13.
5. Zaharia M. et al. Spark: cluster computing with working sets //HotCloud. – 2010. – T. 10. – S. 10-10.
6. Taylor M. TOPCAT: tool for operations on catalogues and tables //Astrophysics Source Code Library. – 2011. – T. 1. – S. 01010.
7. Devereux D. et al. An O (N log M) Algorithm for Catalogue Crossmatching //Astronomical Data Analysis Software and Systems XIV. – 2005. – T. 347. – S. 346.
8. Guttman A. R-trees: a dynamic index structure for spatial searching. – ACM, 1984. – T. 14. – №. 2. – S. 47-57.
9. Li N., Szalay A. CASJobs: A workflow environment designed for large scientific catalogs //2008 Third Workshop on Workflows in Support of Large-Scale Science. – IEEE, 2008. – S. 1-8.
10. Nieto-Santisteban M. A., Thakar A. R., Szalay A. S. Cross-matching very large datasets //National Science and Technology Council (NSTC) NASA Conference. – 2007.
11. Gorski K. M. et al. HEALPix: a framework for high-resolution discretization and fast analysis of data distributed on the sphere //The Astrophysical Journal. – 2005. – T. 622. – №. 2. – S. 759.
12. Zhao Q. et al. A paralleled large-scale astronomical cross-matching function //International Conference on Algorithms and Architectures for Parallel Processing. – Springer Berlin Heidelberg, 2009. – S. 604-614.
13. Pineau F. X., Boch T., Derriere S. Efficient and Scalable Cross-Matching of (Very) Large Catalogs //Astronomical Data Analysis Software and Systems XX. – 2011. – T. 442. – S. 85.
14. CDS X-Match service // URL:http://cdsxmatch.u-strasbg.fr/xmatch.
15. Juric M. Large Survey Database // URL:http://research.majuric.org/trac/wiki/LargeSurveyDatabase.
16. Pineau F. X. et al. Probabilistic multi-catalogue positional cross-match //arXiv preprint arXiv:1609.00818. – 2016.

Поступила 21.10.2016

Об авторах:

Глотов Евгений Сергеевич, магистрант второго года обучения лаборатории Технологий программирования кафедры Автоматизации систем вычислительных комплексов факультета Вычислительной математики и кибернетики Московского государственного университета им. М.В. Ломоносова, glot.unltd@gmail.com;

Герасимов Сергей Валерьевич, инженер факультета Вычислительной математики и кибернетики Московского государственного университета им. М.В. Ломоносова, sergun@gmail.com;

Мещеряков Александр Валерьевич, научный сотрудник Института космических исследований РАН, Казанского (Приволжского) федерального университета, mesch@iki.rssi.ru.