РЕШЕНИЕ ЗАДАЧИ КЛАСТЕРИЗАЦИИ НА ВРЕМЕННЫХ РЯДАХ С ИСПОЛЬЗОВАНИЕМ SPARK ML

А.Н. Вильчицкий[1]
Магистрант кафедры инженерной психологии и эргономики БГУИР,
Республика Беларусь

Е.Н. Унучек[2]
Старший преподаватель кафедры экономической информатики БГУИР,
Республика Беларусь

[1] *Белорусский государственный университет информатики и радиоэлектроники, vilchytski.ales@yandex.by,*
[2] *Белорусский государственный университет информатики и радиоэлектроники, e.unuchek@gmail.com*

This report covers process of applying distributed computing framework Apache Spark and its software library Spark MLLib for time series clustering. Solution of such problem brings ability to improve prediction and optimization of various processes for practical applications. Work covers process of data preparation and storing, practical issues of data processing and interpreting clustering model.

1 Постановка задачи

При решении некоторых практических задач необходим анализ временных данных, представленных в виде временных рядов. Так, например, временными рядами могут быть представлены следующие процессы: потребление вычислительных ресурсов в компьютерной системе, изменение спроса на товары и услуги, изменение погодных условий с течением времени. В сложных системах может наблюдаться множество различных процессов, протекающих как одновременно, так и в разные моменты времени, при этом в некоторых случаях в них можно выделить группы по подобию характеристик.

Рассмотрим в качестве иллюстрации прикладную задачу оптимизации потребления ресурсов вычислительных систем. Разнообразие программных приложений, потребляющих ресурсы систем, как правило, крайне велико, что затрудняет анализ и прогнозирование процессов потребления ресурсов. Исходя из данных о потреблении ресурсов приложениями в течение некоторого периода времени можно выделить группы приложений, процесс выполнения которых имеет схожие характеристики и может быть описан общей моделью. Данная задача может быть решена следующим образом:

1. На основании имеющихся данных о приложениях они могут быть сгруппированы по подобию процесса выполнения – периоды роста и спада потребления ресурсов, время выполнения, скорость изменений и т.д.;

2. В каждой выделенной группе программ может быть смоделирован обобщённый процесс потребления ресурсов, с достаточной точностью отражающий реальные процессы.

В работе рассмотрено решение задачи кластеризации процессов выполнения приложений при помощи следующих инструментов анализа и обработки данных:

1. Apache Spark 1.2 – платформа распределённых вычислений;

2. Spark MLLib – программная библиотека алгоритмов машинного обучения;

3. Apache Cassandra 2 – распределённая нереляционная система управления базами данных.

2 Теоретические сведения

Временной ряд – данные об исследуемом объекте (процессе), представленные в виде конечной последовательности значений наблюдаемой величины η в различные моменты времени:

$$\eta(t_1), \eta(t_2), ..., \eta(t_N), \quad (1)$$

где $t_1, t_2, ..., t_N$ – моменты наблюдений;

N - длина ряда.

Если в каждый момент времени t_i измеряется значение только одной скалярной величины, то временной ряд называют скалярным. Если же одновременно в момент t_i измеряются значения k величин $\eta_1, \ldots, \eta_k$, то ряд называют векторным, т.к. эти величины можно рассматривать как компоненты k-мерного вектора η. Элементы временного ряда (числа или векторы) называют также точками. Порядковый номер точки i называют дискретным временем. Если временные интервалы между последовательными моментами наблюдений t_i одинаковы – $t_i - t_{i-1} = \Delta t$, $i = 2,...,N$, то ряд называют эквидистантным [1].

Определение подобия временных рядов и выделение их групп (кластеров) может производиться при помощи различных алгоритмов кластеризации, в том числе популярного итеративного алгоритма K-средних (*англ.* K-means). Данный алгоритм предназначен для обработки данных, имеющих исчислимый тип и для которых в качестве меры разности применима Евклидова метрика $d(x_i, x'_i)$:

$$d(x_i, x'_i) = \sum_{j=1}^{p} (x_{ij} - x'_{ij})^2 = \|x_i - x'_i\|^2, \quad (2)$$

где x_i, x'_i - сравниваемые точки пространства данных;

p – размерность пространства данных.

Алгоритм K-средних стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров $W(C)$ [2]:

$$W(C) = \sum_{k=1}^{K} N_k \sum_{C(i)=k} \|x_i - \bar{x}_k\|^2 \quad (3)$$

где K – количество кластеров;

N_k – количество точек кластера, такое что среди множества наблюдений I

$$N_k = \sum_{i=1}^{N} I(C(i) = k)$$

C(i) – номер некоторого кластера;

$\bar{x}_k$ – центральная точка кластера.

При решении задач кластеризации временных рядов может возникнуть проблема большой размерности данных, актуальная для длинных серий значений при вычислении Евклидовой метрики. Существуют различные техники снижения размерности временных данных, одна из которых – дискретное преобразование Фурье (*англ.* discrete Fourier transform, DFT), определение которому можно дать следующим образом. Будем считать, что временной ряд эквидистантный с чётным числом точек N, интервал выборки Δt, $a = t_1$, $b = a + N \cdot \Delta t = t_N + \Delta t$. Можно показать, что исходный ряд $\{\eta(t_i)\}_{i=1}^{N}$ может быть однозначно представлен во все моменты наблюдений в виде суммы спектральных составляющих:

$$\eta(t_i) = a_0 + \sum_{k=1}^{N/2} a_k \cos(k\omega t_i) + \sum_{k=1}^{N/2-1} b_k \sin(k\omega t_i), \quad i = 1,2,...,N \quad (4)$$

где a_k, b_k – коэффициенты спектральных составляющих;

ω - основная частота $\omega = \dfrac{2\pi}{N\Delta t}$.

Коэффициенты спектральных составляющих вычисляются по следующей формуле:

$$a_0 = \frac{1}{N}\sum_{i=1}^{N} \eta_i, \quad a_{N/2} = \frac{1}{N}\sum_{i=1}^{N} (-1)^i \eta_i \quad (4.1)$$

$$a_k = \frac{2}{N}\sum_{i=1}^{N} \eta_i \cos(k\omega t_i), \quad k = 1,2,...,N/2 \quad (4.2)$$

$$b_k = \frac{2}{N}\sum_{i=1}^{N} \eta_i \sin(k\omega t_i), \quad k = 1,2,...,N/2 \quad (4.3)$$

Формулы (4.1) - (4.3) называются прямым дискретным преобразованием Фурье (*англ.* discrete Fourier transform, DFT). Евклидова метрика, применяемая к исходному и преобразованному в ряд Фурье рядам, будет пропорциональна [1]. В различных работах можно найти варианты применения DFT, улучшающие пара-

метры преобразованных данных с точки зрения кластерного анализа. Основные способы снижения размерности данных при работе с рядами Фурье – сглаживание, ограничение спектра по значимым (верхним или нижним) частотам, а также выборка частот с учётом их веса [3, 4].

3 Применение Apache Spark и Cassandra для обработки временных данных

Обработка больших объёмов данных, требуемых для построения качественных статистических моделей кластеризации, требует эффективных и производительных средств вычисления. На сегодняшний день существуют эффективные распределённые системы вычислений, позволяющих обеспечить параллельную обработку данных. Apache Spark – распределённая система вычислений общего назначения. Она предоставляет высокоуровневые программные интерфейсы на языках Java, Scala и Python. В состав системы входят программные библиотеки, содержащие алгоритмы машинного обучения (библиотека Spark MLLib) и линейной алгебры (библиотека Breeze). Модель вычислений на платформе Spark строится вокруг двух классов операций – преобразований и действий (*англ*. transformations and actions), выполняемых над данными, сгруппированными в особые коллекции – Resilient Distributed Dataset (RDD) – неизменяемые распределённые коллекции элементов любого сериализуемого типа, которые могут быть обработаны параллельно. В качестве примеров трансформаций RDD можно назвать операции фильтрации (filter), преобразования (map) и объединения (join), а примерами действий над RDD – операции сохранения (persist), кэширования в памяти (cache) и итерации (foreach). В качестве источника данных для RDD могут выступать файлы, базы данных, очереди сообщений и др. [5].

При обработке больших объёмов данных, для которых требуется высокая скорость обработки и не требуется обеспечение транзакционности, можно применять хранилища под управлением нереляционных системы управления базами данных (СУБД). Apache Cassandra – это распределённая СУБД, относящаяся к классу NoSQL-систем и рассчитанная на создание масштабируемых и надёжных хранилищ больших массивов данных [6].

4 Описание процесса решения задачи

В качестве входных данных для решения задачи кластеризации временных рядов были взяты данные о процессах потребления вычислительных ресурсов компьютерной системы различными приложениями, собранные с интервалом 30 секунд (всего 429241 отсчёт). Задача кластеризации временных рядов решается в следующем порядке:

1. Подготовка данных для анализа при помощи Spark – объединение данных из разрозненных источников во временные ряды, соответствующие отдельным процессам;

2. Сохранение подготовленных данных в Cassandra – опциональный шаг для ускорения итеративного процесса анализа данных;

3. Применение алгоритма кластеризации – преобразование временных рядов при помощи дискретного преобразования Фурье, кластеризация при помощи реализации алгоритма К-средних в Spark MLLib;

4. Анализ и интерпретация результатов.

Рассмотрим структуру временных данных. Данные загружаются в 2 таблицы. Таблица описания процессов содержит сводные сведения о выполняемых приложениях: время, идентификатор приложения, имя приложения. Таблица измерений процессов содержит сведения о потреблении ресурсов каждым приложением в разные моменты времени: время, идентификатор приложения, индекс потребления ресурсов. Объединим имеющиеся таблицы для получения информации о каждом процессе в отдельные моменты времени, что упростит дальнейшую интерпретацию данных. Также преобразуем структуру хранения для оптимального доступа к информации при пакетной обработке при помощи Spark. Полученная объединённая таблица процессов будет содержать все поля таблиц описания процессов и измерений процессов, объединённые по следующим полям: время, идентификатор приложения.

Рассмотрим, как можно выполнить эффективное соединение (*англ*. join) двух таблиц Cassandra при помощи Apache Spark. Операции объединения в Spark могут быть выполнены различными способами, в том числе при помощи встроенных функций RDD. Однако, повысить эффективность операции соединения можно за счёт устранения фазы обмена данными между узлами (*англ*. shuffling). В случае с Cassandra это возможно в том случае, если таблицы имеют общий ключ распределения данных (*англ*. partition key). В решаемой задаче таким ключом является сочетание полей «время» и «идентификатор приложения» таблиц описания и измерения процессов. Это означает, что записи в Cassandra, имеющие одинаковые значения данных полей, будут физически расположены на одном узле базы данных, а значит, для их объединения не потребуется shuffling. Поэтому, вместо встроенной функции RDD *join* применим более эффективную функцию *joinWithCassandraTable*, учитывающую физическое расположение данных в Cassandra [7]. На выходе операции получим новый RDD, представляющий объединённую таблицу процессов.

Подготовка данных для пакетной обработки и кластерного анализа. Для эффективного хранения и обработки временных рядов может быть применена схема таблицы Cassandra с использованием списочного типа данных (list), представленная на рисунке 1. Таблица содержит информацию о приложении – его идентификатор, название и списки моментов времени, в которые проводились измерения, а

также результаты измерений индекса потребления ресурсов в соответствующие моменты. Назовём данную таблицу таблицей временных рядов.

Partition Key

Идентификатор приложения	Имя приложения	Время	Индекс потребления ресурсов
1	java	List(1, 2, 3)	List(100, 80, 60)
2	python	List(1, 2, 3, 4, 5)	List(10, 20, 10, 0, 1)
3	scala	List(1, 2, 3, 4, 5)	List(1, 5, 10, 20, 1)

Рис. 1. Схема таблицы временных рядов в базе данных Cassandra, адаптированная для пакетной обработки.

Заполнение данной таблицы возможно при помощи последовательности следующих трансформаций, применяемых к объединённой таблице процессов:

1. *map* – операция преобразует строку объединённой таблицы процессов в кортеж из двух элементов – текстовой строки, состоящей из идентификатора приложения, а также самой строки таблицы.

2. *combineByKey* – трансформация, определяемая тремя функциями:

2.1. Создание из одной строки объединённой таблицы процессов одной строки таблицы временных рядов;

2.2. Добавление значений времени и соответствующих измерений из строки объединённой таблицы процессов к строке таблицы временных рядов;

2.3. Комбинация строк таблицы временных рядов, имеющих общий идентификатор приложения, при которой выполняется слияние списков моментов времени и соответствующих измерений. Важно отметить, что порядок поступления данных на вход трансформации не определён, а потому необходимо находить место вставки элементов в списки таким образом, чтобы элементы были упорядочены по значениям моментов времени.

3. *map* – операция перехода от промежуточного формата данных к строкам таблицы временных рядов.

После выполнения описанных выше трансформаций можно выполнить операцию сохранения данных в Cassandra. Используя полученную таблицу временных рядов можно быстро и эффективно подготовить данные для кластерного анализа.

Следует отметить, что на практике полученные временные ряды могут иметь ряд свойств, затрудняющих дальнейший анализ:

1. Временные ряды могут иметь разную длину. Необходимо принять меры к их выравниванию – выделить временное окно или дополнить значения в малых рядах нолями до длины максимально длинных рядов.

2. Ряды могут содержать большое количество значений, что затруднит расчёт модели кластеризации. Необходимо уменьшить размерность, для чего может быть

применено дискретное преобразование Фурье, вейвлет-преобразование и другие методы.

3. В рядах могут быть пропущенные значения и выбросы. Их можно либо восстановить при помощи различных статистических методов, либо отбросить как непригодные для анализа.

В рамках решаемой задачи выполняется дополнение рядов до максимальной длины нолями и дискретное преобразование Фурье. Будем полагать, что эти ряды не имеют существенных пропусков и выбросов. Для определения максимальной длины ряда применим к таблице временных рядов функцию Spark RDD *max*, которая параметризуется функцией сравнения элементов по длине временных рядов. Получив максимальную длину ряда выполним трансформацию *map*, по следующему алгоритму: если временные ряды имеют длину, меньшую максимальной – дополним их нолями до вычисленной ранее максимальной длины. Над полученными списками измерений выполним дискретное преобразование Фурье при помощи функции библиотеки Breeze *fourierTr.dvDouble1DFFT*. Данная функция возвращает вектор комплексных чисел, из которого для дальнейшего анализа оставим первую половину элементов и возьмём действительную часть каждого из них. Таким образом, на выходе преобразования получим основные составляющие ряда Фурье, отражающие амплитудные характеристики временных рядов [1], что сокращает размерность временных данных вдвое.

Полученные в результате преобразований векторы могут быть переданы на вход алгоритма кластеризации K-means в Spark MLLib. Данная реализация алгоритма выбирает начальные центры кластеров случайным образом и параметризуется следующими величинами:

Количество кластеров в анализируемых данных;

Количество итераций алгоритма кластеризации;

Количество запусков алгоритма кластеризации (будет возвращён результат наилучшего запуска).

Алгоритм K-средних в Spark MLLib возвращает набор векторов и меток кластеров, к которым они относятся. Поскольку оптимальные параметры кластеризации заранее неизвестны, можно выполнить несколько запусков алгоритма с различными параметрами и выбрать наилучший результат по критерию минимизации квадратичной ошибки в кластерах. Для её расчёта Spark MLLib предоставляет функцию *computeCost*.

Кластеризованные при помощи K-means временные ряды свяжем с имеющейся информацией из предметной области, что упростит интерпретацию результатов. Это можно сделать при помощи функции Spark MLLib *predict*, которая для точки (вектора значений) возвращает метку кластера, к которой она относится. Сопоставив таким образом исходные данные о программных приложениях, взятые из таб-

лицы временных рядов, и метки их кластеров можно получить модель кластеризации данных, которая проще в интерпретации с точки зрения предметной области. Spark позволяет сохранить итоговые данные как в базу данных Cassandra при помощи операции RDD *writeToCassandraTable*, так в локальную или распределённую файловую систему при помощи операций *saveAsTextFile*, *saveAsHadoopFile* и др.

5 Описание результатов

Проанализируем один из вариантов модели кластеризации исходных данных, в котором выделено 4 кластера из 5847 временных рядов. Распределение рядов по кластерам приведено в таблице 1.

Таблица 1 – распределение временных рядов по кластерам.

Номер кластера	Количество временных рядов
1	5038
2	534
3	52
4	223
Итого	5847

При помощи метода главных компонент построим графическую интерпретацию распределения элементов в группы, приведённую на рисунке 2. В ней в качестве характеристики каждой точки кластера выделены две компоненты [2, 8].

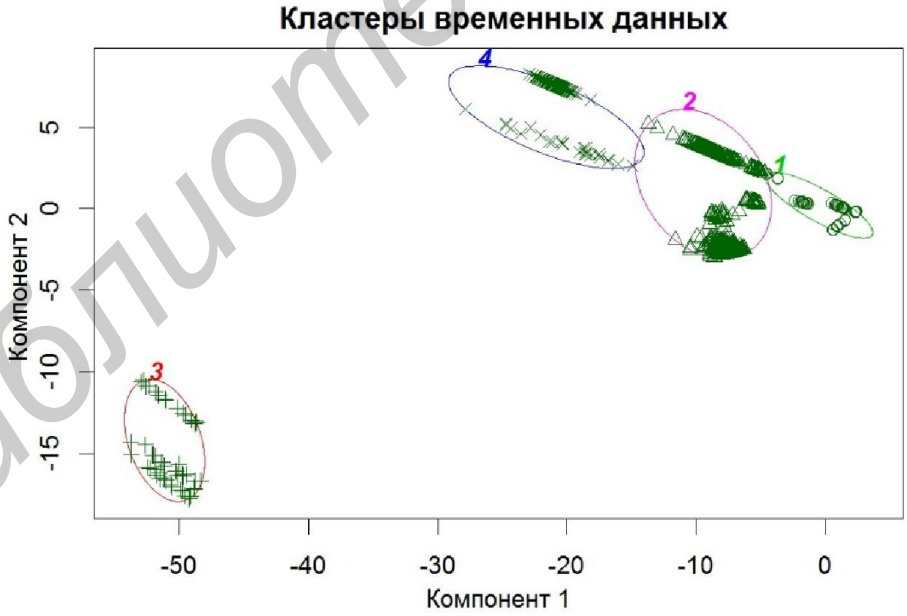

Рис. 2. Графическая интерпретация модели кластеризации.

Данная графическая интерпретация с достоверностью 87% отражает взаимное расположение сгруппированных рядов, которые достаточно точно распознаны алгоритмом К-средних.

Рассмотрим несколько случайно выбранных из каждого кластера рядов, чтобы получить сведения об их основных свойствах. Достаточно наглядна в этом случае графическая интерпретация значений отсчётов в различные моменты времени, приведённая на рисунках 3 - 6. На основании полученных данных можно судить о характере процессов, сгруппированных в кластеры.

Рис. 3. Зависимость отсчётов от времени случайно выбранных временных рядов в первом кластере.

В первом кластере сгруппированы временные ряды с малой флуктуацией отсчётов и большим временем наблюдения. С точки зрения предметной области это фоновые программы, выполняющиеся постоянно и характеризующиеся постоянным потреблением ресурсов.

Рис. 4. Зависимость отсчётов от времени случайно выбранных временных рядов во втором кластере.

Во втором кластере сгруппированы процессы с малым временем жизни и скачкообразным ростом значений отсчётов, соответствующие ресурсоёмким служебным приложениям, выполняемым периодически.

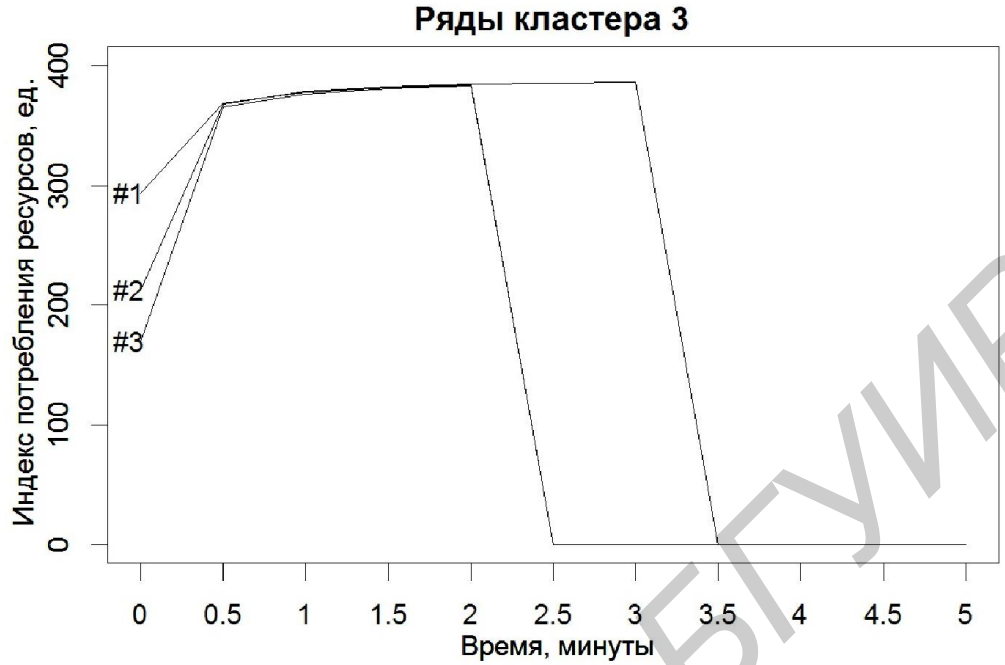

Рис. 5. Зависимость отсчётов от времени случайно выбранных временных рядов в третьем кластере.

В третий кластер сгруппированы процессы со средним временем жизни и большим потреблением ресурсов, которые относятся к пользовательским приложениям, запускаемым для выполнения ресурсоёмких задач.

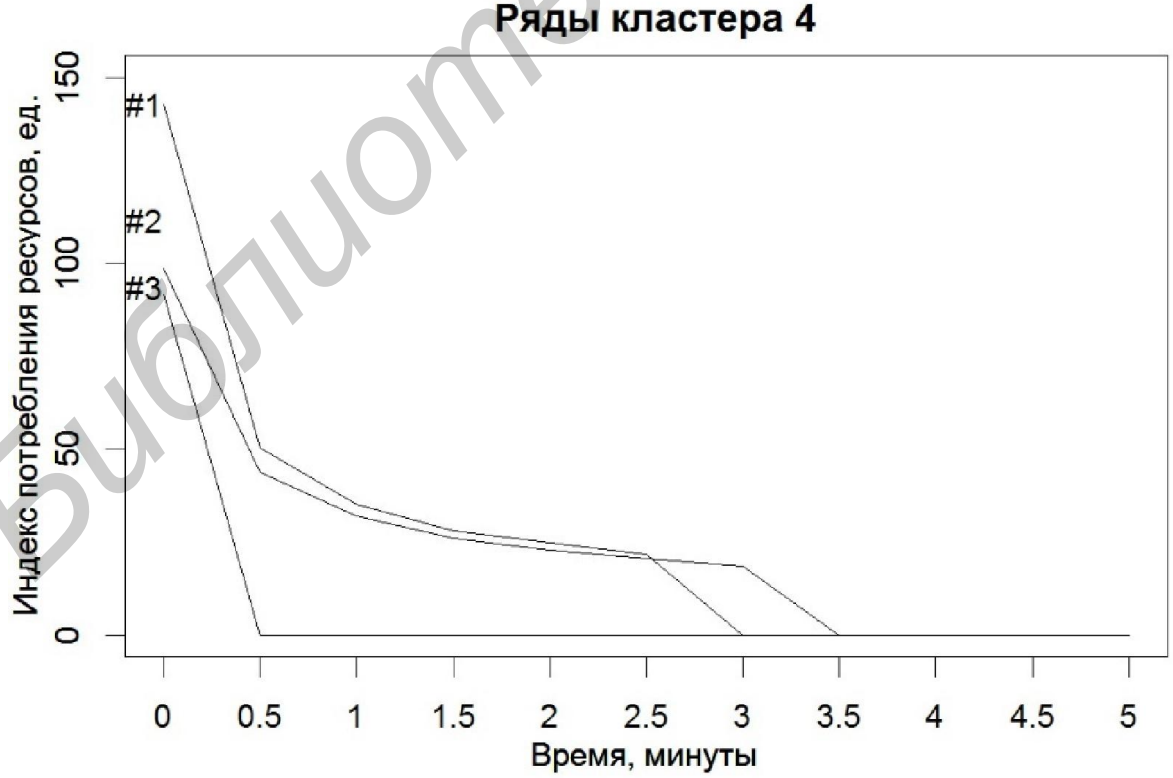

Рис. 6. Зависимость отсчётов от времени случайно выбранных временных рядов в четвёртом кластере.

В четвёртом кластере размещены процессы со средним временем жизни, которые относятся к пользовательским приложениям со средним потреблением ресурсов.

Таким образом, среди всего многообразия процессов, протекающих в исследуемой системе, были выделены 4 группы, что соответствует разделению приложений по характеру потребления ресурсов в зависимости от времени выполнения.

6 Оценка полученных результатов

Полученная при помощи Spark MLLib модель К-средних описывает группы программных приложений, выполняемых в вычислительной системе, по подобию процесса потребления ресурсов. Полученное распределение приложений по группам рационально с точки зрения предметной области. Таким образом, полученная модель может быть применена при решении задач оптимизации и прогнозирования потребления ресурсов системы. В дальнейшем, данное решение задачи кластеризации временных рядов может быть эффективно масштабировано для анализа больших объёмов данных за счёт возможностей платформы Apache Spark и СУБД Cassandra.

Однако, следует указать на недостатки применённых методов и возможную неточность полученных результатов. Применение дискретного преобразования Фурье при анализе временных рядов имеет следующие недостатки [1]:

1. Единичное преобразование Фурье не позволяет выявить изменения частотного состава процесса со временем;

2. Если процесс содержит частоты, большие и кратные частоте измерений, это может внести искажения в низкочастотную часть спектра в виде наложения частот;

Алгоритм кластеризации К-средних имеет следующие недостатки [2]:

1. Качество построенной модели зависит от выбора центров кластеров, оптимальный выбор которых, как правило, заранее неизвестен;

2. Для алгоритма необходимо задавать количество кластеров, которое, как правило, заранее неизвестно;

3. Евклидова метрика, применяемая в алгоритме, не всегда позволяет корректно сопоставить временные ряды в случае временных смещений и изменений их частотного состава.

Среди альтернатив использованным при решении задачи кластеризации на временных данных алгоритмам можно выделить следующие:

1. Алгоритм иерархической кластеризации (*англ*. hierarchical clustering), применённый к временным рядам, прошедшим процедуру динамической трансформации временной шкалы (*англ*. dynamic time warping, DTW), что позволяет применить более рациональную метрику подобия временных рядов [2];

2. Применение алгоритма К-средних к временным рядам, подвергнутым дискретному вейвлет-преобразованию (*англ*. discrete wavelet transform, DWT), что позволяет обнаруживать в спектре процесса изменения частотного состава и имеет ряд других преимуществ перед дискретным преобразованием Фурье [1].

Следует отметить, что приведённые выше альтернативные алгоритмы не реализованы в Apache Spark и Spark ML версии 1.2, что затрудняет их применение на данной платформе.

Литература

1. Безручко Б.П., Смирнов Д.А. Математическое моделирование и хаотические временные ряды. / Б.П. Безручко, Д.А. Смирнов – Саратов: ГосУНЦ «Колледж», 2005. 320 с

2. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. Data Mining, Inference, and Prediction. Second Edition. – Springer, 2010. 745p.

3. Moerchen F. Time series feature extraction for data mining using DWT and DFT. *Philipps-University Marburg* – 2005, 31p.

4. Keogh E. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Department of Information and Computer Science University of California* – 2000, 19p.

5. Apache Spark online documentation [Электронный ресурс]. – Режим доступа: https://spark.apache.org/docs/1.2.2/ – Дата доступа: 18.05.2015.

6. Apache Cassandra online documentation [Электронный ресурс]. – Режим доступа: http://docs.datastax.com/en/cassandra/2.0 – Дата доступа: 18.05.2015.

7. DataStax spark-cassandra-connector repository and documentation [Электронный ресурс] – режим доступа: https://github.com/datastax/spark-cassandra-connector/tree/v1.2.1/doc – Дата доступа: 18.05.2015.

8. Bivariate Cluster Plot (clusplot) in R [Электронный ресурс] – режим доступа: https://stat.ethz.ch/R-manual/R-devel/library/cluster/html/clusplot.default.html – Дата доступа: 18.05.2015.