Министерство образования и науки Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«Московский физико-технический институт
(государственный университет)»

Факультет управления и прикладной математики

Кафедра информатики

Управление дисковой памятью системы хранения данных LHCb на основе прогноза популярности данных

Выпускная квалификационная работа
(магистерская диссертация)

Направление подготовки: 03.04.01 Прикладные математика и физика

Подготовил:
студент 973 группы — Гущин Михаил Иванович

Научный руководитель:
к.ф.-м.н., доцент — Устюжанин Андрей Евгеньевич

Москва 2015

Содержание

1 Введение

LHCb коллаборация - это один из четырех главных экспериментов на Большом Адронном Коллайдере в ЦЕРН. Детекторы LHCb и моделирование физических процессов генерируют огромный объем данных ежегодно. Данные хранятся на жестких дисках и магнитных лентах. Диски используются для хранения данных, которые физики используют для своих текущих исследований. Жесткие диски намного быстрее магнитных лент и значительно дороже их, в результате объем дискового пространства существенно ограничен. Поэтому очень важно определить какие файлы нужно держать на дисках, а какие хранить в виде архивов на магнитных лентах.

В данной работе представлена рекомендательная система для управления дисковой памятью систем хранения данных LHCb. Система сконструированна так, чобы отбирать файлы, которые будут востребованы в будущем, и поэтому должны храниться на жестких дисках. Входными даными системы являются история обращений к файлам и их метаданные.

Рекомендательная система состоит из трех модулей. Первый модуль - Data Popularity Estimator. Это модуль использует методы машинного обучения и входные данные системы, чтобы предсказать популярность файлов. Популярность файлов выражает вероятность того, что файл будет использован в будущем. На основании популярности данных можно определить какие файлы могут быть удалены с диска.

Второй модуль - Data Intensity Predictor. Этот модуль нужен, чтобы спрогнозировать интенсивность обращений к файлам. Для прогноза используются методы анализа временных рядов и регрессионного анализа. Прогноз строится на истории обращений к данным.

Третий модуль - Data Placement Optimizer. Этот модуль использует предсказанные популярность данных и интенсивность обращений, чтобы определить какие файлы должны остаться на диске и сколько копий они должны иметь. Для этого минимизируется функция потерь. Функция потерь представляет все требования, которые мы предъявляем к распределению данных в системе хранения данных LHCb.

Все три модуля подробно описаны в следующих секциях. В секции результатов приведено сравнение нашей рекомендательной системы и Last Recently Used (LRU) алгоритма.

2 Существующие решения

2.1 SSD-оптимизированное распределение нагрузки с адаптивным обучением и классификацией в HPC средах

Авторы работы *SSD-optimized workload placement with adaptive learning and classification in HPC environments*[3] представляют решение для гибридной HDD + SSD системы хранения данных. В такой системе объем SSD дисков ограничен, и нет возможность использовать только SSD диски. Поэтому, на SSD дисках необходимо хранить только наиболее популярные данные. Остальные данные хранятся на HDD дисках.

2.1.1 Введение

Представленная система использует адаптивную классификацию файлов по популярности для перемещения данных между медленными HDD дисками и быстрыми SSD дисками и для выполнения пользовательских требований к операциям загрузки/выгрузки данных. Разработанный алгоритм использует ряд предположений. Во-

первых, авторы предполагают, что система содержит как медленные HDD, так и быстрые SSD диски. Во-вторых, пользователи могут иметь свои требования к распределению своих данных в системе. Например, определенные файлы должны храниться на HDD дисках в трех копиях, быть доступными определенный отрезок времени и другие.

На основании сделанных предположений, предложенное решение состоит из двух частей. Во-первых, авторы используют основанную на цепях Маркова модель классификации, чтобы предсказать какие файлы будут интенсивно использоваться в будущем, используя историю обращений к данным. Во-вторых, разработан основанный на линейном программировании алгоритм для распределения данных, который учитывает требования пользователей к пропускной способности и надежности системы.

2.1.2 Постановка задачи

Имеется набор устройств для хранения данных, представленные HDD и SSD дисками. Задача авторов работы заключалась в том, чтобы найти такое распределение данных в системе, которое:

- Удовлетворяет требованиям пользователей к системе,
- Оптимизирует пропускную способность операций ввода/вывода для HPC приложений.

Также задача имеет свои особенности:

- Шаблон доступа к данным в будущем не известен,
- Требования пользователей к системе могут сильно отличаться и постоянно меняться.

2.1.3 Классификация нагрузки на основе цепей Маркова

Описанный авторами метод классификации содержит следующие шаги. Во-первых, предполагается, что известна история обращений к каждому объекту данных. В реальности, может быть не возможным хранить всю историю обращений к данным, а только за определенный недавний промежуток времени. Во-вторых, частота обращений для каждого объекта данных моделируется с помощью дискретных цепей Маркова, в которой каждое состояние представляет определенный промежуток значений частоты обращений. В-третьих, вычисляя стационарное распределение цепи Маркова, авторы вычисляют вероятность того, что частота обращений к объекту данных будет лежать в определенном интервале значений в будущем. Затем, каждый объект ранжируется с помощью взвешенной суммы стационарного распределения, где веса определяются интервалом значений частоты обращений к объекту для каждого состояния цепи Маркова. Чем выше значение ранга, тем выше выигрыш от перемещения объекта на SSD диск. Эти шаги подробнее описываются далее.

1) История обращений к файлам: Рисунок 1 демонстрирует частоту обращений к данным (частота включает операции чтения и записи) для файла за последний месяц. Ось X на рисунке представляет отрезок времени в один месяц, разделенный на 720 промежутков (1 промежуток равен 1 часу). Ось Y выражает число обращений к файлу за каждый промежуток времени. Представленная история обращений используется для построения цепи Маркова, чтобы предсказать частоту обращений в будущем. Как показано на рисунке 1, только выделенная окном история обращений

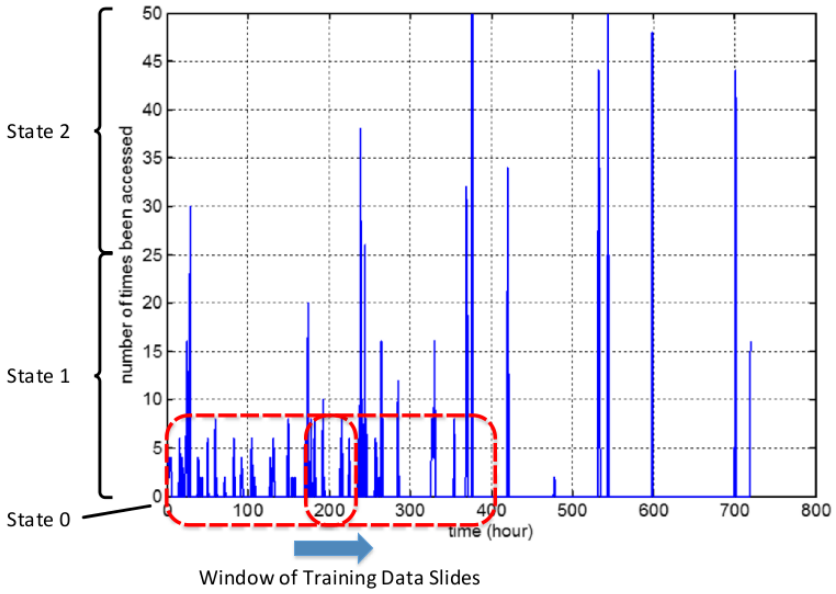

Рис. 1: История обращений к файлу.

используется для обучения модели. Перемещая окно со временем можно реализовать онлайн прогноз частоты обращений к файлу.

2) Предсказательная модель: Используя историю обращений для каждого файла, строится модель цепи Маркова, чтобы предсказать частоту обращений в будущем для каждого объекта. Для начала, необходимо определить число состояний цепи Маркова, и какой интервал частоты обращений должен соответствовать каждому состоянию. Например, как показанно на рисунке 1, если максимальное число обращений в течение одного промежутка времени равно 50, то 50 можно разделить на два промежутка и построить цепь Маркова, которая будет иметь три состояния: 0, $(0, 25]$ and $(25, 50]$ соответственно. Если в течение одного промежутка времени не будет ни одного обращения к файлу, тогда цепь Маркова будет находиться в состоянии 0. Если число обращений будет больше 0, но меньше 25, то тогда цепь Маркова будет в состоянии 1, и так далее. Диаграмма переходов для цепи Маркова изображена на рисунке 2.

Затем, история обращений преобразуется в последовательность состояний цепи Маркова используя определенные интервалы значений числа обращений для каждого состояния. Например, последовательность состояний для истории обращений, показанной на рисунке 2 $1, 1, 1, 1, 1, 0, 0...$ На основании такой последовательности состояний можно определить вероятности переходов между каждыми двумя состояниями цепи Маркова и построить матрицу перехода как показано ниже:

$$\mathbf{T} = \begin{pmatrix} p_{00} & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ p_{20} & p_{21} & p_{22} \end{pmatrix} \tag{1}$$

В соответствии со свойствами цепей Маркова получаем:

$$\lim_{n\to\infty} \mathbf{T}^n = \begin{pmatrix} \pi_0 & \pi_1 & \pi_2 \\ \pi_0 & \pi_1 & \pi_2 \\ \pi_0 & \pi_1 & \pi_2 \end{pmatrix} \tag{2}$$

где $\pi{=}[\pi_0, \pi_1, \pi_2]$ называется стационарным распределением цепи Маркова. Также, π можно найти с помощью левого собственного вектора $\mathbf{E}$ с единичной нормой

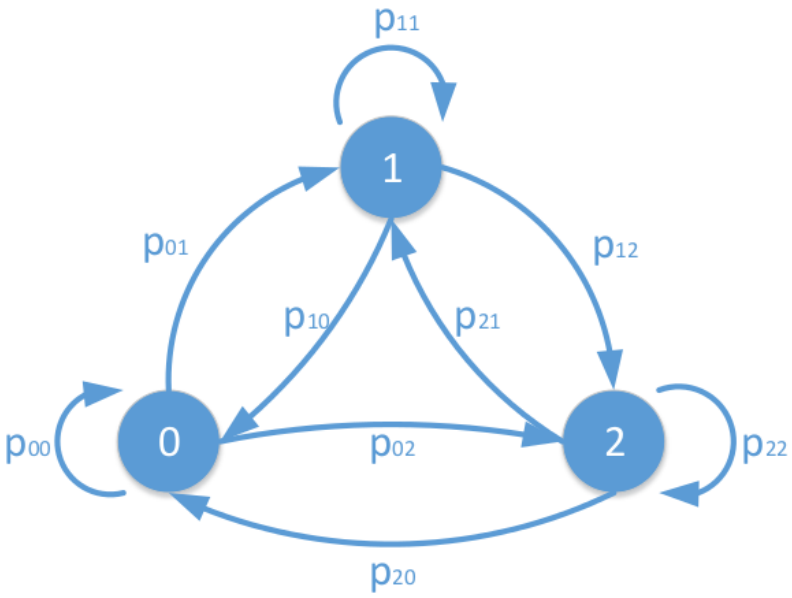

Рис. 2: Диаграмма переходов для цепи Маркова.

для матрицы перехода **T**:

$$\pi = \frac{\mathbf{E}}{\sum_i e_i} \tag{3}$$

где e_i - i-ый элемент собственного вектора **E**. Так как стационарное распределение π отражает вероятность того, что каждое состояние цепи Маркова будет активно в будущем, это свойство может быть использованно, чтобы предсказать число обращений к каждому файлу.

Используя предсказанные частоты обращений можно ранжировать файлы так, чтобы ранг показывал какие объекты должны храниться или должны быть удалены с SSD дисков. Однако, должно быть учтено то, что даже если стационарное распределение покажет, что состояние 1 будет активным с большей вероятностью, чем состояние 2, состояние 2 выражает большее число обращений к файлу. Поэтому, используется взвешенная сумма стационарного распределения, чтобы ранжировать файлы по их значимости. Веса выбираются пропорциональными интервалам числа обращений для каждого состояния цепи Маркова. Например, если стационарное распределение π=$[0.31, 0.56, 0.13]$, и взяты веса $[0, 10, 20]$ для трех состояний цепи, то ранг для файла вычисляется как $rank_{obj} = 0.31 \times 0 + 0.56 \times 10 + 0.13 \times 20 = 8.2$. Тогда ранг объекта можно использовать для определения оптимального распределения данных в системе хранения данных.

2.1.4 Распределение данных

Так как файлы ранжированны по значимости, следующий шаг - найти такое распределение данных, которое минимизирует время задержки обращения к данным и будет удовлетворять требованиям пользователей к системе. Авторы работы полагают, что требования пользователей можно выразить через уравнения и неравенства. Например, используя обозначения из таблицы 1, требование, чтобы минимальное число копий для i-го файла равнялось 3 можно записать как $cp_i > 3$.

Таблица 1: Обозначение символов.

N	Общее число дисков
M	Общее число файлов
cs_i	Емкость i-го диска
ds_i	Размер i-го объекта данных
f_i	Предсказанная частота обращений для i-го файла
b_{ij}	Пропускная способность соединения между i-ым и j-ым дисками
at_i	Пропускная способность i-го диска
e_{ij}	Хранится ли i-ый файл на j-ом диске (0 или 1)
cp_i	Минимальное число копий для файла i

Задачу поиска оптимального распределения данных авторы записывают в виде задачи оптимизации:

Поиск максимума выражения:

$$\sum_{i \in M} f_i \times max[j \in N, at_j \times e_{ij}] \tag{4}$$

с ограничениями:

$$\sum_{j \in N} e_{ij} = cp_i, i \in M, \tag{5}$$

$$\sum_{i:e_{ij}=1} ds_i \leq cs_j, j \in N, \tag{6}$$

В этом примере выражение 4 выражает желание авторов распределить файлы между дисками так, чтобы максимизировать взвешенную сумму пропускных способностей дисков, где веса - частота обращений.

Первое ограничение, выражение 5, отображает то, что число копий файлов задается пользователем. Второе ограничение - ни один диск не может содержать больше данных, чем его емкость.

Таким же образом можно выразить более сложные требования пользователей.

2.1.5 Результаты

Анализируя историю обращений к различным файлам, авторы выяснили, что все объекты можно разделить на две категории по истории обращений к ним. К первой категории относятся файлы с постоянной частотой обращений к ним. Файлы из этой категории часто используются на протяжении всего срока существования файла без существенной разницы между минимальным и максимальным числом обращений к ним. На рисунке 3 изображен пример файла из 1 категории. Алгоритмы машинного обучения, в частности цепи Маркова, позволяют достичь высокой точности для таких объектов. Ко второй категории относятся файлы, который используются очень редко. Однако, число обращений за один промежуток времени для таких фалов может принимать большие значения. Пример файла из второй категории изображен на рисунке 4. Как сообщают авторы работы, для таких файлов очень трудно сделать достаточно точный прогноз числа обращений, используя любой алгоритм машинного обучения, включая цепи Маркова.

Авторы сравнивают среднюю пропускную способность на чтение файлов, которые можно достичь применяя их предсказательную модель на основе цепей Маркова

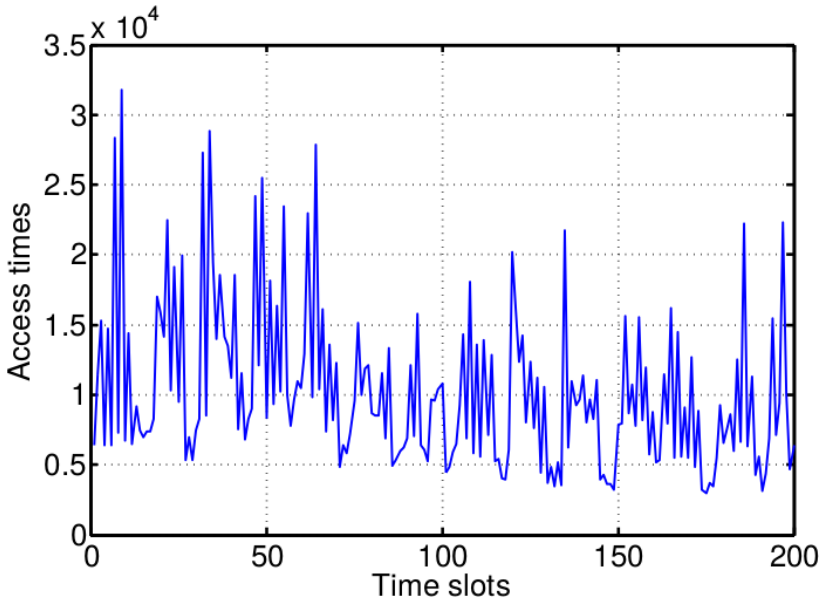

Рис. 3: История обращений 1го типа.

и применяя модель случайного выбора. Модель случайного выбора заключается в том, что случайно выбираются объекты данных (файлов), которые будут храниться на SSD дисках. Так как число SSD дисков ограничено, авторы рассчитывают среднюю пропускную способность на чтение для систем с долей SSD дисков от 2,5% до 50%. Более того, авторы полагают, что пропускная способность SSD дисков на чтение равна 550 МБ/с и для HDD дисков - 120 МБ/с. Рисунок 5 показывает, что представленный авторами работы метод демонстрирует свою эффективность по сравнению с моделью случайного выбора.

2.2 Система перераспределения данных для распределенной системы хранения данных ATLAS

В работе *A Popularity-Based Prediction and Data Redistribution Tool for ATLAS Distributed Data Management*[4,5] авторы представляют систему для предсказания популярности данных в системах с большим объемом данных, таких как система распределенного хранения данных ATLAS. Используя полученный прогноз популярности, возможно перераспределить данные в системе для уменьшения времени задержки для задач, которые используют эти данные. Прогноз популярности производится по истории обращений к данным с применением искусственных нейронных сетей[1,2].

2.2.1 Введение

ATLAS - один из четырех основных экспериментов Большого Адронного Коллайдера. В распределенной системе хранения данных ATLAS хранятся огромные объемы данных с детектора и данных моделирования физических процессов. На момент написания работы система содержала более 150 ПБ экспериментальных данных, которые распределены на более чем 150 сайтах по всему миру.

В своей работе авторы представляют новый способ автоматически и динамически удалять и добавлять новый копии данных в системе в соответствии с будущей популярностью данных. Данный способ реализован в трех частях. В первой части анализируется история обращений к данным, чтобы сделать предсказать число воз-

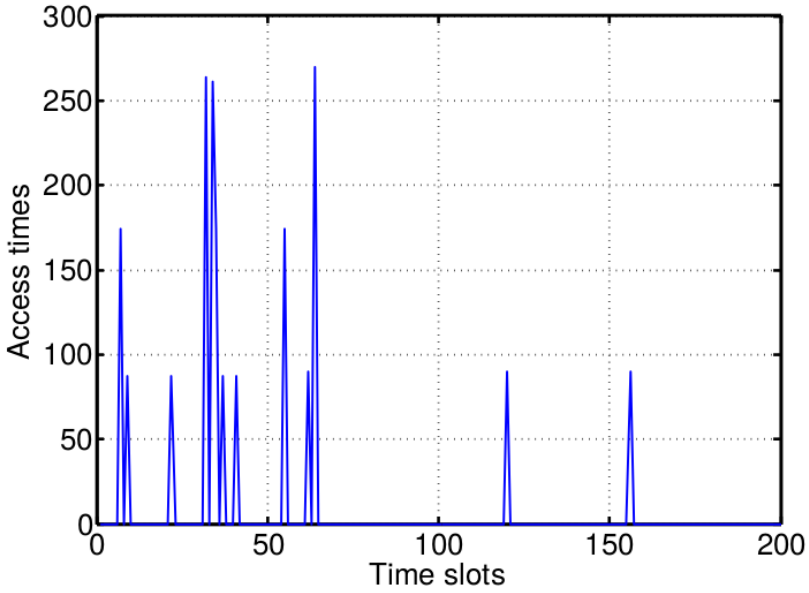

Рис. 4: История обращений 2го типа.

можных обращений в ближайшем будущем. Затем, используя полученный прогноз, данные в системе перераспределяются путем добавления или удаления копий файлов. Как утверждают авторы, очень трудно определить преимущества таких методов на живых системах, где шаблон нагрузки на систему никогда не повторяется. Поэтому, третья чать их работы посвящена моделированию системы распределенного хранения данных с повторяющимся шаблоном нагрузки для определения качетсва их метода.

До применения предложенного авторами метода, распределение данных в системе носило статический характер, что приводило в большому числу невостребованных копий файлов в системе. Идея предложенного метода в том, чтобы автоматически определять невостребованные файлы и динамически освобождать место для более популярных файлов. Делается общее предположение, что польза от большего числа копий фалов заключается в том, что: 1) уменьшается время ожидания пользователей, перед тем как из задачи получат данные для дальнейшей работы, 2) ресурсы системы используются более эффективно.

2.2.2 Прогноз популярности

Чтобы проводить распределение данных в системе лучше, необходимы знания о будущем числе обращений к данным. Т.е. необходимо сделать прогноз числа обращений в будущем. Предполагается, что историю поведения пользователей можно использовать для прогноза популярности данных в будущем. Существует множество способов сделать необходимы прогноз. Авторы работы рассматривают два таких способа.

1) Общая концепция. Система хранения данных записывает обращения всех пользователей к файлам. Эти данные могут быть использованы для прогноза. На практике прогноз делается в недельном базисе. Такие временные рамки были выбраны подходящими для перераспределения данных в системе. Однако, меньшие или большие отрезки времени так же могут быть использованы. Входными данными для прогноза популярности является история обращений к файлам за последние n недель ($A_{1,2,\ldots,n}$). На выходе получаем число обращений к файлу за следующую неделю (A_{n+1}).

Один из наиболее простых методов - статический прогноз. Предположение за-

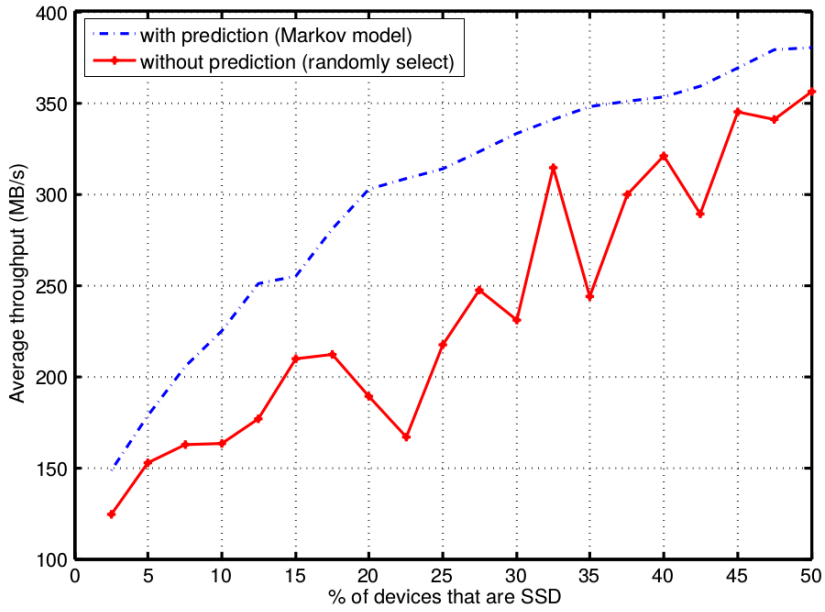

Рис. 5: Средняя пропускная способность на чтение от доли SSD дисков в системе.

ключается в том, что число обращений к файлу остается постоянным в течение следующей недели, т.е. $A_{n+1} = A_n$. Этом метод будет использован авторами работы для сравнения с их методом.

2) Искусственные нейронные сети. Другой, более сложный метод, который учитывает историю обращений к файлу за предыдущие недели - это обучение искусственных нейронных сетей[1,2] (ИНС). Общая идея заключается в том, чтобы построить нейронную сеть с n входными нейронами и одним выходным нейроном. Каждому входному нейрону соответствует число обращений за одну из последних n недель истории обращений к файлу, и выходной нейрон выдает прогноз числа обращений для $n + 1$ недели. Пример изображен на рисунке 6.

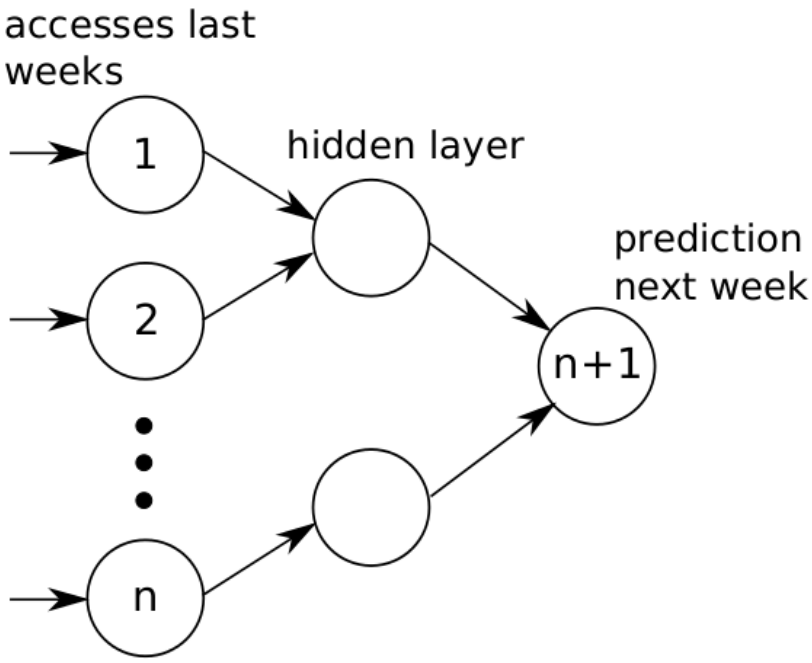

Рис. 6: Пример нейронной сети.

Выше была описана общая концепция метода, но для практического применения необходимо учесть ряд особенностей. Файлы, которые хранятся в системе, отличаются разными типами групп пользователей, которые этими файлами пользуются. Соответственно, каждая группа пользователей имеет свой шаблон поведения. По утверждению авторов, каждый шаблон поведения может сильно меняться для разных типов данных, и объединение всех этих шаблонов делает невозможным хороший прогноз с применением нейронных сетей. Еще одним шагом, который поможет нейроной сети обнаружить шаблон поведения, является отбор данных перед обучением модели. Некоторые файлы используются очень редко и сильно не регулярно. По мнению авторов, для таких файлов метод статического прогноза дает хорошие результаты. Модель нейронных сетей применяется для файлов с числом обращений выше некоторого порогового значения.

После того как данные были отобраны, обучалась нейронная сеть. Для обучения модели необходимы данные для обучения и их метки. Идея заключается в том, чтобы использовать $1, 2..., n-1$ недели для обучения и n-ую неделю в качестве меток данных обучения. После того, как модель нейронной сети будет обучена, ее можно будет использовать для прогноза популярности файлов. Так как входные данные сдвинуты, т.е. теперь $2, 3, ..., n$ недели являются входными данными, а выходными данными будет прогноз числа обращений к файлам за $n + 1$-ую неделю.

2.2.3 Перераспределение данных

Алгоритм перераспределения данных состоит из двух частей. Первая часть - освобождений места в системе и создание новых копий. Вторая часть - процесс удаления, который определяет сколько байт должны быть удалены для новых копий и как эффективно использовать ресурсы.

1) Создание копий. Алгоритм перераспределения данных основан на прогнозе числа обращений к файлам. Метод, представленный авторами, использует прогноз числа обращений для равномерного распределения копий файлов между имеющимися ресурсами. Каждый сайт, где хранятся данные, содержит определенное число слотов для одновременного выполнения нескольких задач. Если все эти слоты заняты, то новой задаче приходится ждать своей очереди. При увеличении числа реплик файла, увеличивается число слотов для задач, которые работают с этим файлом, что приводить к уменьшению общего времени ожидания.

Для предложенного метода равномерного распределения нагрузки использовалась метрика равная аккумулированному числу обращений на один слот одного сайта. Аккумулированное число обращений для каждого сайта вычислялось как сумма числа обращений к каждой копии на сайте. Так как число слотов для сайтов различны, то аккумулированное число обращений нормируется на число слотов сайта.

2) Удаление копий. При удалении копий в процессе перераспределения данных необходимо учитывать две вещи. Во-первых, для каждого файла должно оставаться минимально число копий. Минимальное число копий зависит от типа данных. Это требование связано с тем, чтобы ни один файл не был полностью удален и не произошло потери данных из-за возможных сбоев системы. Во-вторых, число копий может быть уменьшено только для тех файлов, которые были непопулярными в течение последних недель.

Перераспределение. Для перераспределения данных требуются следующие входный данные:

- **Текущее распределение данных:** Каталог данных с распределеним копий для каждого файла.

- **Текущая конфигурация файлов:** Конфигурация сайта содержит информацию об объеме дисков и числе слотов.
- **История обращений к файлам:** Понедельное число обращений к файлам за прошлые недели.
- **Прогноз числа обращений:** Прогноз числа обращений к файлам на следующей неделе.
- **Максимальное число байт:** Максимальное число байт, которое алгоритму можно использовать для перераспределения данных в системе. Т.е. не может быть удалено или добавлено больше чем заданное число байт.

Первым шагом перед перераспределением данных является вычисление аккумулированных чисел обращений к уже доступным копиям. Аккумулированное число обращений для каждого сайта - сумма числа обращений к каждой копии на сайте. Перераспределение начинается с пробега по файлам с прогноза. Файлы сортируются в порядке убывания числа обращений. Для каждого файла создается список сайтов для возможного создания копии. Этот список строится по возрастанию отношения аккумулированного числа обращений к числу слотов. Все сайты, которые уже содержат копии данного файла исключаются из списка. Следующим шагом, новую копию файла пытаются записать на первый сайт в списке. Если места на сайте не достаточно, включается алгоритм удаления копий, который удаляет непопулярные копии до тех пор, пока не освободиться достаточно метса. Если нет возможности освободить достаточно места, алгоритм переходит к другому сайту. В конце, если места на сайте достаточно, копия успешно добавляется, и алгоритм продолжает работу с другим файлом. Процесс продолжается до тех пор, пока не будут созданы дополнительные копии для всех популярных файлов, или пока не достигнет максимального числа байт.

Рисунок 7 демонстрирует работу алгоритма на простом примере.

2.2.4 Результаты

Одна из метрик, которую использовали авторы для сравнения алгоритмов - среднее время ожидания задачи. Рисунки 8, 9 показывают зависимость среднего времени ожидания для статической модели прогноза и модели нейронной сети в течение двух недель от максимального размера перераспределения данных. Графики демонстрируют тренд на увеличение преимущества от применения представленного авторами метода при увеличении максимального размера перераспределения данных. Значительного уменьшения времени ожидания удается достичь на отрезке от 500 ТБ до 1000 ТБ. После 1000 ТБ большего уменьшения достичь не удается. Как утверждают авторы, при максимальном размере перераспределения данных в 1000 ТБ, используя статическую модель прогноза, удается снизить время ожидания с 4 до 3 часов. Используя модель нейронной сети, этих же результатов можно достичь уже при 500 ТБ.

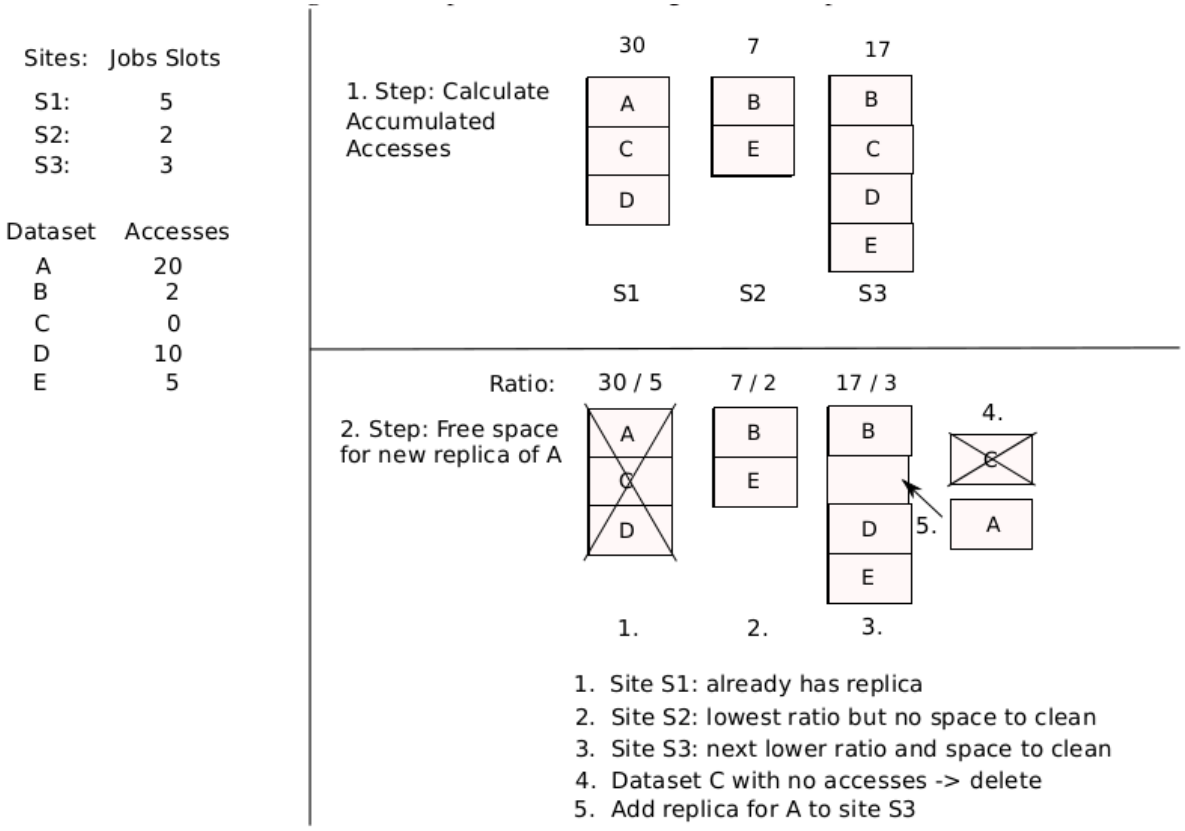

Рис. 7: Пример работы алгоритма перераспределения данных.

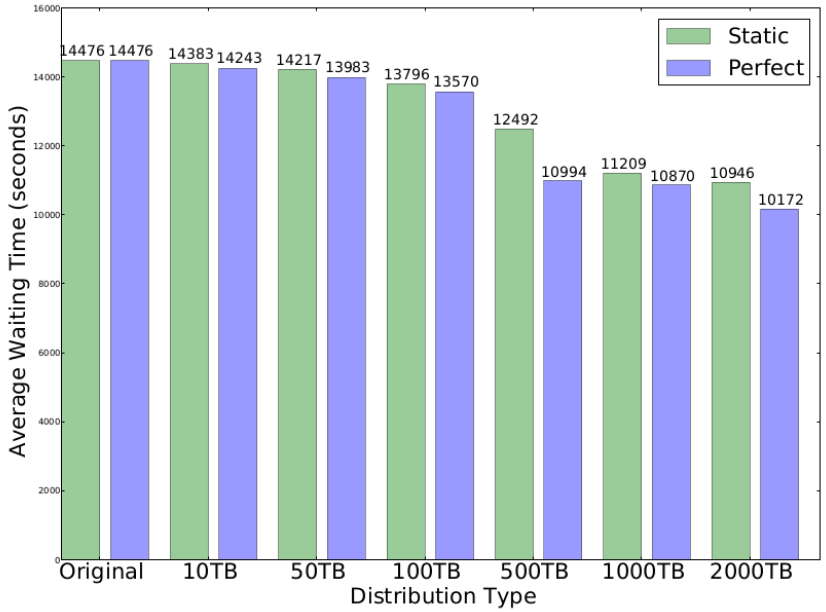

Рис. 8: Среднее время ожидания для одной недели (04.11-10.11).

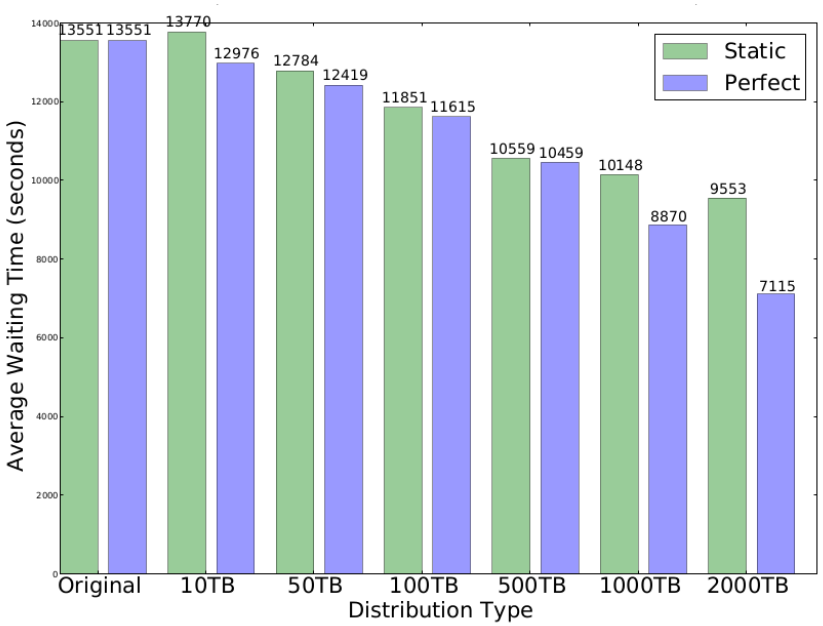

Рис. 9: Среднее время ожидания для одной недели (18.11-24.11).

Результаты для второй недели отличаются. Связь между временем ожидания и максимальным размером перераспределения данных более линейная. Также, для этой недели максимальный выигрыш достигается при 2000 ТБ, где время ожидания уменьшается с 3.75 до 2 часов. Используя статическую модель прогноза, авторы смогли уменьшить среднее время ожидания до 2.6 часов. При малых размерах перераспределенных данных обе модели показывают схожие результаты. Однако, уже при 1000 ТБ модель нейронной сети показывает свое преимущество.

2.3 Другие работы

Кроме работ по применению популярности данных для оптимизации управления системами хранения данных, существует ряд работ по прогнозу популярности других объектов. Искусственные нейронные сети[1], линейные модели классификации[1], регрессионные модели[1] и анализ временных рядов[2] широко используются для прогноза популярности комментариев пользователей[6,7,8], контента в системах IPTV[9], видео в социальных сетях[10,12], YouTube[11].

3 Особенность работы

Особенность данной работы в том, что история обращения к файлам сильно разрежена. Т.е. временной ряд, представляющий историю обращений имеет большое число нулевых значений. Большая часть файлов в данной работе используются редко, однако большое число раз. Такие файлы попадают во вторую категорию в работе [3] и не используются для обучения нейронной сети в работах [4,5]. Такие методы как искусственные нейронные сети[1], ARMA[2], ARIMA[2] и другие слишком сложны для этих данных и демонстрируют плохие результаты. Как будет показанно ниже, мы вычисляем дополнительные признаки для каждого файла, которые характеризуют особенности их истории обращений. Эти признаки используются для классификации файлов на популярные и непопулярные.

Каждый файл имеет метаданные, которые содержат информацию о типе данных, что содержится в этом файле и другую информацию. Использование этих данных в качестве признаков для решения задачи классификации позволяет избежать стадии разделения данных по типу и типу пользователей, как это было описанно в работе [4].

4 Постановка задачи

Разработать рекомендательную систему для системы хранения данных LHCb. Система должна выполнять следующие функции:

- Рекомендовать файлы для хранения на жестких дисках.
- Рекомендовать число копий файлов на дисках.
- Допускать минимум ошибок (т.е. когда файл был удален с диска, но затем его использовали).
- Экономно использовать дисковое пространство.
- Не приводить к значительному увеличению времени доступа к данным.

5 Решение задачи

5.1 Входные данные

История обращений к файлам и метаданные используются как входные данные рекомендательной системы. В данной работе мы использовали понедельное число обращений к файлам собранные за два последних года. История обращений к файлам представлена как временные ряды из 104 точек. Каждая точка представляет число обращений к файлу за одну неделю.

Метаданные файлов содержат следующую информацию: источник данных, конфигурация детектора, тип файла, тип данных (Монте Карло или реальные данные), тип события, время создания файла, время первого обращения, время последнего обращения, размер одной копии, общий размер файла на диске, число копий на диске и некоторые другие.

5.2 Модуль Data Popularity Estimator

Модуль Data Popularity Estimator использует классификатор для вычисления популярности данных. Классификатор - это алгоритм машинного обучения с учителем[1] и состоит из нескольких шагов. Следующие подсекции описывают каждый шаг определения популярности данных.

5.2.1 Разметка файлов

Так как классификатор - это алгоритм машинного обучения с учителем, то каждый файл должен быть размечен на популярный и непопулярный. Временные ряды истории обращений к файлам сильно разрежены, поэтому последние 26 недель истории обращений используются для разметки данных. Если файл не используется в последние 26 недель, мы помечаем его как непопулярный и даем метку 1. В противном

случае, мы помечаем файл как популярный и даем ему метку 0. Эти метки определяют класс файла (0 для популярных, 1 для непопулярных). На рисунках 10 и 11 представлены файлы из каждого класса.

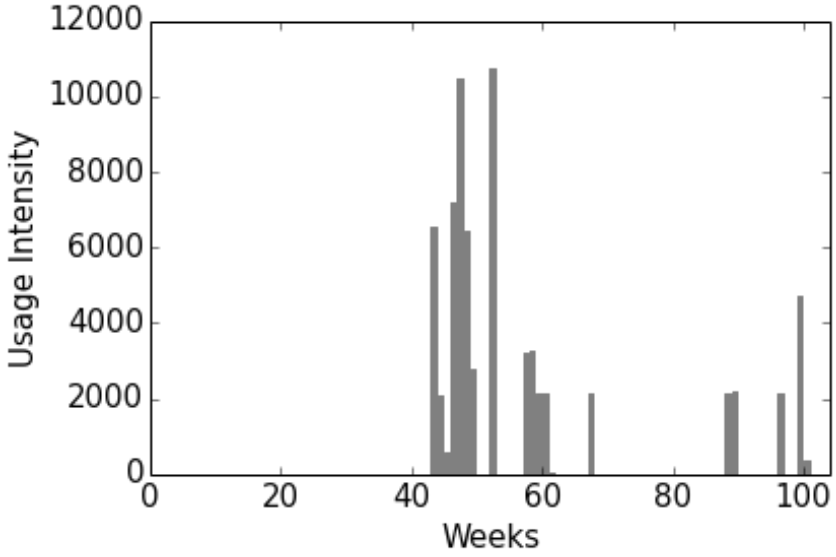

Рис. 10: Временной ряд с меткой 0.

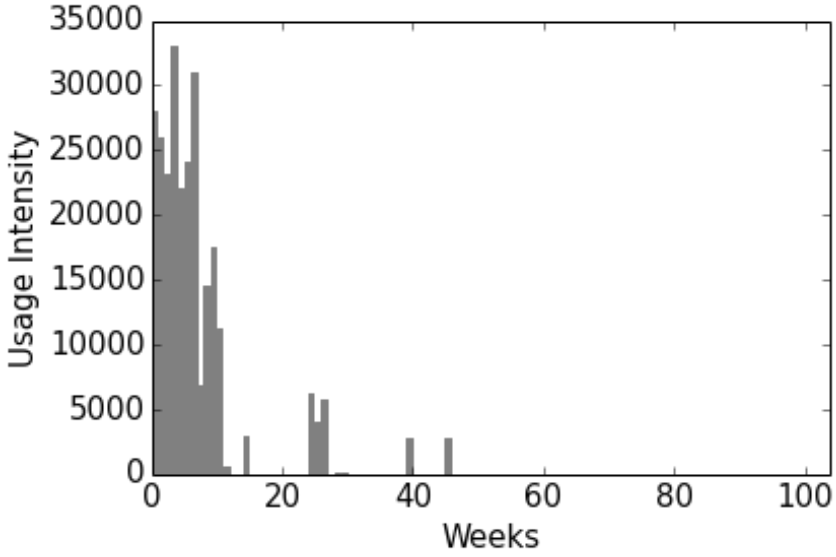

Рис. 11: Временной ряд с меткой 1.

5.2.2 Предобработка данных

Метаданные файлов используются как входные признаки для классификатора. Были вычислены новые признаки, которые использовались вместе с существующими. Эти новые признаки описывают форму временного ряда для истории обращений к файлу. Так как последние 26 недель истории обращений используются для разметки файлов, то для вычисления новых признаков используются только первые 78 недель. Имена новых признаков: *nb_peaks*, *last_zeros*, *inter_max*, *inter_mean*, *inter_std*, *inter_rel*, *mass_center*, *mass_center_sqrt*, *mass_moment* and *r_moment*.

Nb_ peaks - число недель, в течение которых к файлу обращались. *Last_ zeros* - число недель с тех пор, как к файлу в последний раз обращались. *Inter_ max*, *inter_ mean*, *inter_ std* выражают максимальное значение, среднее значение и стандартное отклонение числа недель между последовательными неделями с ненулевым числом обращений. *Inter_ rel* - отношение *inter_ std* к *inter_ mean*. *Mass_ center* выражает центр масс временного ряда файла, в котором масса - это число обращений к файлу за каждую неделю, а координата - номер недели. *Mass_ center_ sqrt*, *mass_ moment* и *r_ moment* аналогичны *mass_ center*, но масса и координата берутся с разными степенями.

Эти новые признаки существенно увеличивают качество классификации.

5.2.3 Обучение классификатора

Новые признаки, метаданные файлов и их метки используются для обучения классификатора. В качестве классификатора используется градиентный бустинг[1] за его высокое качество классификации, быструю работу и отсутствие переобучения. Для обучения классификатора использовался метод перекрестной проверки с 10 частями (k-fold cross-validation)[1]. Все файлы были разбиты на 10 частей. Классификатор обучался на 9 частях данных, а затем использовался для предсказания вероятности получить метку 1 для 10-ой части данных. На рисунке 12 изображено распределение вероятности для каждого из классов.

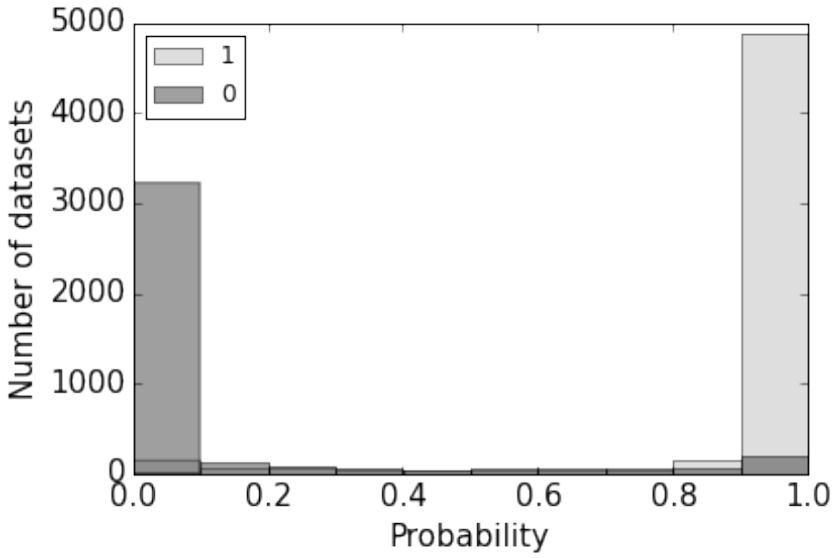

Рис. 12: Распределение вероятности получить метку 1 для каждого класса файлов.

5.2.4 Определение популярности данных

Описанная выше вероятность преобразуется в популярность так, чтобы популярность для класса с меткой 1 была равномерно распределена. Чем ближе популярность к 1, тем выше вероятность того, что файл не будет использован в будущем. Т.е. вычисленная популярность определяет антипопулярность файла. Такой странный выбор связан с тем, чтобы первые на удаление файлы имели большую величину метрики. Рисунок 13 демонстрирует распределение популярности для каждого класса файлов.

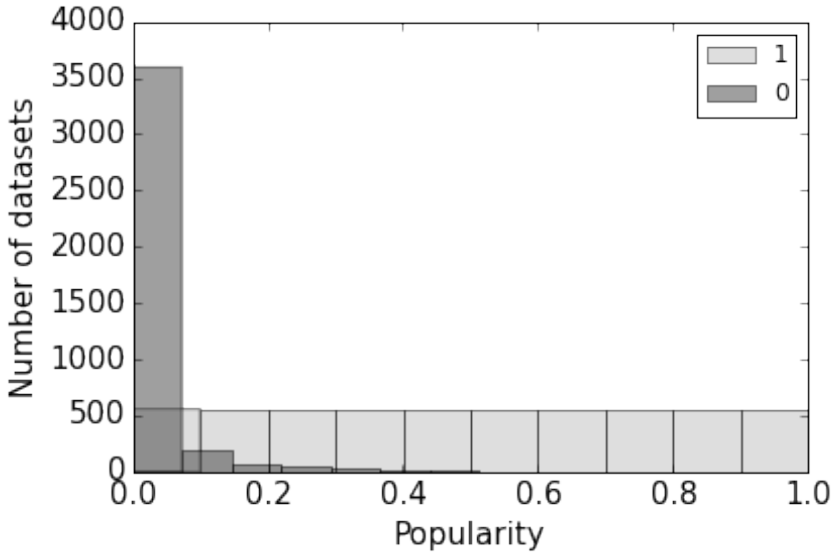

Рис. 13: Распределение популярности для каждого класса файлов.

5.3 Модуль Data Intensity Predictor

Как было написано выше, популярность данных выражает вероятность того, что файл будет бесполезным в будущем. Второй важной характеристикой является интенсивность обращения к файлу. Существует множество алгоритмов анализа временных рядов[2] для предсказания будущих значений временных рядов. Так как большая часть временных рядов в данной работе сильно разряжена, сложные параметрические методы, такие как полиномиальная регрессия[1], авторегрессия[1,2], ARMA[2], ARIMA[2], искусственные нейронные сети[1] и другие не подходят для этой задачи. В этой секции описывается применение двух непараметрических моделей[1] для прогноза интенсивностей обращений к файлам. Этими методами являются метод ядерного сглаживания Надарая-Ватсона[1] и метод скользящего среднего[1,2].

5.3.1 Метод ядерного сглаживания Надарая-Ватсона

Пусть точки $(x_1, y_1), (x_2, y_2), ..., (x_l, y_l)$ представляют временной ряд и $X^l = \{x_1, x_2, ..., x_l\}$. Тогда, формула ядерного сглаживания *Надарая-Ватсона*[1]:

$$\hat{y}_h(x; X^l) = \frac{\sum_{i=1}^{l} y_i K(\frac{\rho(x,x_i)}{h})}{\sum_{i=1}^{l} K(\frac{\rho(x,x_i)}{h})}, \tag{7}$$

где

$\hat{y}_h(x; X^l)$ - значение временного ряда в точке x после ядерного сглаживания на значениях X^l,
$K(\frac{\rho(x,x_i)}{h}) = exp(-\frac{(x-x_i)^2}{2h^2})$ - RFB ядро сглаживания,
h - ширина окна сглаживания.

Для выбора оптимального окна сглаживания был выбран метод *Leave-One-Out*[1]:

$$LOO(h, X^l) = \sum_{i=1}^{l} (\hat{y}_h(x_i; X^l \setminus \{x_i\}) - y_i)^2 \mapsto \min_h \tag{8}$$

Формула ядерного сглаживания *Надарая-Ватсона*[1] с LOO[1] оптимизацией ширины окна сглаживания применяется на временных рядах истории обращений к фай-

лам. Максимальная ширина окна сглаживания была взята в 30 недель. На рисунке 14 изображен пример временного ряда после процедуры ядерного сглаживания.

5.3.2 Метод скользящего среднего

На следующем шаге вычисляется скользящее среднее[1,2] для дополнительного сглаживания временных рядов. Пусть точки $(x_1, y_1), (x_2, y_2), ..., (x_l, y_l)$ представляют временной ряд после ядерного сглаживания. Тогда, скользящее среднее вычисляется следующим образом:

$$\hat{y}_k = \frac{\sum_{i=k-w}^{k} y_i}{w} \tag{9}$$

где w - ширина скользящего окна.

Ширина скользящего окна выбрана так, чтобы 90% всех временных рядов с одинаковыми значениями nb_peaks имели значения $inter\_max$ меньше либо равными ширине окна.

Скользящее среднее в момент x_i представляет значение интенсивности обращения к файлу в этот момент. Самый простой способ предсказать будущее значение временного ряда - это взять в качестве будущего значения ряда значение в последней точке наблюдения (статическая модель прогноза в работе [4]). Пример скользящего среднего и предсказанного значения интенсивности обращений к файлу представлен на рисунке 14.

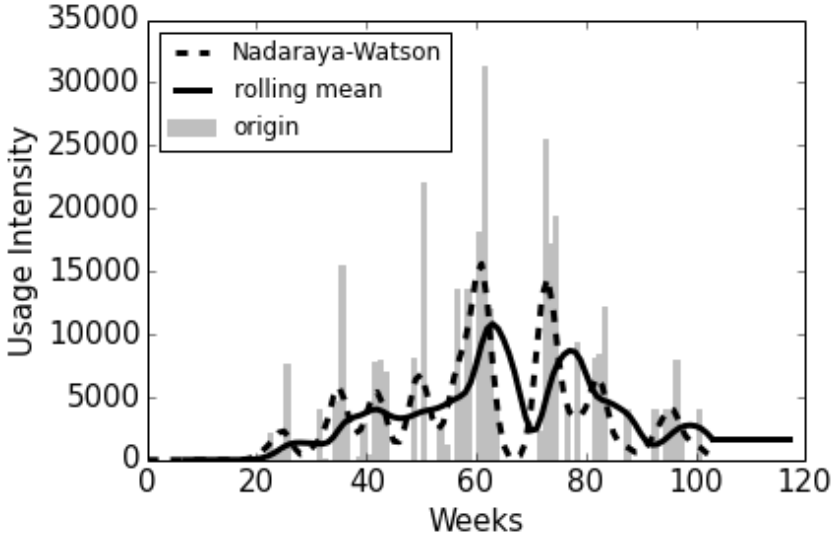

Рис. 14: Пример временного ряда после применения формулы ядерного сглаживания *Надарая-Ватсона* и вычисления скользящего среднего.

5.4 Модуль Data Placement Optimizer

В этой секции описывается способ определения того, какие файлы должны храниться на жестких дисках и сколько копий они должны иметь, используя популярность и предсказанное значение интенсивности обращений к файлам. По причине того, что дисковое пространство более дорогое, чем пространство магнитных лент, мы хотим занимать как можно меньше дискового пространства. С другой стороны, крайне нежелательно удалить с жесткого диска файлы, которые будут востребованы в будущем. Более того, мы хотим хранить наиболее востребованные файлы с большим числом копий, чтобы уменьшить среднее время доступа к данным.

Описанные выше требования отображены в следующей функции потерь:

$$L = C_{disk} \sum_{i}^{n} S_i(Rp_i + \alpha \frac{I_i}{Rp_i})\delta_i + C_{tape} \sum_{i}^{n} S_i(1 - \delta_i) + C_{miss} \sum_{i}^{n} S_i m_i, \qquad (10)$$

C_{disk} - цена 1 Гб жесткого диска,
C_{tape} - цена 1 Гб магнитных лент,
C_{miss} - цена восстановления 1 Гб данных с магнитных лент на диск,
α - штраф за малое число копий,
S_i - размер одной копии i файла,
Rp_i - число копий i файла,
I_i - предсказанная интенсивность обращения к i файлу;
δ_i - равно 1 если i файл на диске, 0 - иначе;
m_i - равно 1 если i файл был восстановлен с магнитных лент на диск.

Первое слагаемое в функции потерь выражает цену хранения файлов на жестких дисках. Второе слагаемое - цена хранения файлов на магнитных лентах. Последнее слагаемое равно цене ошибок, когда файл был удален с жесткого диска, но затем был использован.

Выражение в скобках в первом слагаемом функции потерь используется, чтобы определить оптимальное число копий файлов на дисках, используя предсказанные значения интенсивностей обращений к временным рядам. Оптимальное число копий файла с предсказанной интенсивностью обращений I_i и для значения α определяется выражением:

$$Rp_{i\_optimal} = \sqrt{\alpha I_i}, \qquad (11)$$

На рисунке 15 представлена зависимость оптимального числа копий для файла от его значения предсказанной интенсивности обращений и для различных значений альфа. Например, предположим, что предсказанная интенсивность $I = 10$ обращений в неделю и $\alpha = 0.5$. Тогда $Rp_{optimal} = \sqrt{\alpha I} = \sqrt{0.5 * 10} = 2.24 \approx 2$ копий.

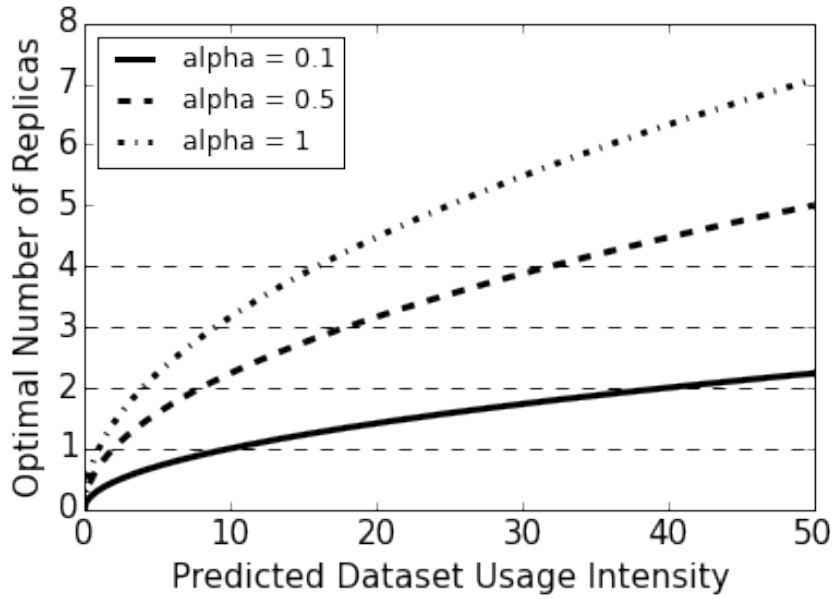

Рис. 15: Зависимость оптимального числа копий файла (Rp) от предсказанного значения интенсивности обращения к файлу (I) и α.

Значение δ_i в функции потерь зависит от порогового значения популярности файла. Файл со значением популярности равным или большим, чем пороговое значение удаляются с жесткого диска ($\delta_i = 0$). Значение m_i - это произведение $1 - \delta_i$ и метки класса i^{th} файла.

Оптимизация функции потерь состоит в том, чтобы найти такие значения порогового значения популярности данных и оптимальные значения копий файлов, которые доставляют функции потерь минимум.

6 Сравнение алгоритмов

6.1 LRU алгоритм

В этой работе мы приводим сравнение нашей рекомендательной системы и Last Recently Used (LRU) алгоритма. LRU алгоритм смотрит на последние наблюдения в истории обращений к файлу и принимает решение о том, какие файлы должны быть удалены с диска. В данной работе временные ряды первых 78 недель истории обращений к файлам использовались как входные данные для алгоритма. Последние 26 недель истории использовались для измерения качества алгоритма. Тогда, если файл не использовался в течение последних N недель (с $(78 - N)$-ой по 78-ую), то этот файл удаляют с диска. Число копий не меняется, по сравнению с первоначальным числом копий.

6.2 Время доступа к данным

Время доступа к данным оцениваем временем загрузки всех файлов всеми пользователями:

$$T = \sum_{i=1}^{n} I_i^* S_i t_{disk} \alpha(Rp_i)\delta_i + \sum_{i=1}^{n} (K_{tape} + S_i t_{tape}) m_i + \sum_{i=1}^{n} I_i^* S_i t_{disk} m_i \qquad (12)$$

где $\alpha(Rp_i) = 0.05 + \frac{1}{Rp_i}$
t_{disk} - среднее время загрузки 1 Гб данных с диска,
t_{tape} - среднее время загрузки 1 Гб данных с магнитных лент на диск,
K_{tape} - константное время, необходимое для восстановления файла с магнитной ленты на диск,
I_i^* - среднее число обращений (загрузок) к файлу за неделю,
S_i - размер одной копии i файла,
Rp_i - число копий i файла,
δ_i - рано 1, если i файл на диске, иначе - 0,
m_i (ошибки классификации) - равно 1, если i файл был восстановлен с ленты на диск.

Первое слагаемое выражения для времени загрузки - время загрузки всех файлов с жестких дисков всеми пользователями. Второе слагаемое выражает время, необходимо для восстановления тех файлов с магнитных лент, которые были удалены с диска из-за ошибки алгоритма. Третье слагаемое - время, необходимое для загрузки восстановленных файлов всеми пользователями. Первые 78 недель истории обращений к файлам используются как входные данные алгоритмов. Последние 26 недель используются для измерения качества алгоритмов и для определения числа обращений (загрузок) к файлам.

6.3 Результаты

Файлы, которые были созданы и впервые использованы ранее 78-ой недели используются для сравнения алгоритмов. 7375 файлов участвовали в сравнении. Следующие значения параметров использовались для оптимизации функции потерь: $C_{disk} = 100$, $C_{tape} = 1$, $C_{miss} = 2000$. Для выражения времени загрузки использовались следующие значения параметров: $t_{disk} = 0.1$ часа/Гб, $t_{tape} = 3$ часа/Гб and $K_{tape} = 24$ часов. Значения этих параметров отражают идеи того, что объем жестких дисков ограничен, восстановление файла с магнитной ленты на диск требует большого количества времени, число ошибок алгоритма должно быть минимальным.

Таблицы 2 и 3 отображают результаты сравнения нашей рекомендательной системы с максимальным числом копий файла равным 4 и LRU алгоритма. *Отношение времени доступа* - это отношение времени доступа (загрузки) к файлам после применения алгоритма к первоначальному времени доступа. Колонка *Экономия места* показывает сколько места на жестких дисках можно сэкономить с помощью алгоритма. Колонка *Число ошибок* показывает число файлов, которые были удалены с жесткого диска, но затем были использованы.

Оба алгоритма позволяют сохранить сравнимый объем дискового пространства. Однако, наша рекомендательная система допускает гораздо меньше ошибок. Также таблицы показывают, что наша система с максимальным числом копий для файла равным 4 позволяет незначительно снизить время доступа к данным.

Таблица 2: Результаты для LRU алгоритма.

N	Отношение времени доступа	Экономия места, %	Число ошибок
1	1.33	63	1973
2	1.28	58	1659
5	1.4	50	1357
10	1.11	44	966
15	1.07	38	635
20	1.03	33	370
25	1.02	30	193

Таблица 3: Результаты нашей системы с максимальным числом копий равным 4.

Альфа	Отношение времени доступа	Экономия места, %	Число ошибок
0	3.35	71	9
0.01	0.99	46	9
0.05	0.96	34	9
0.1	0.96	30	9
0.5	0.96	23	9
1	0.96	19	9
2	0.96	16	9

Таблица 4 показывает, что при максимальном числе копий равным 7 наша система помогает сэкономить до 40% объема жесткого диска и снизить время загрузки файлов на 30%.

Таблица 4: Результаты нашей системы с максимальным числом копий равным 7.

Альфа	Отношение времени доступа	Экономия места, %	Число ошибок
0	3.35	71	8
0.001	1.03	57	8
0.005	0.72	40	8
0.01	0.68	34	8
0.05	0.63	11	8
0.1	0.62	1	8

Анализ выполнен с применением Reproducible Experiment Platform[15] - платформы для решения задач анализа данных.

7 Библиотека

Представленная здесь рекомендательная система реализована в виде библиотеки на языке Python. Библиотека называется *datapop*[13] и может быть загружена из репозитория *https://github.com/hushchyn-mikhail/DataPopularity* с подробной инструкцией по использованию.

8 Сервис

Также разработан веб-сервис нашей рекомендательной системы. Сервис *datapopserv*[13] написан на языке Python с использованием библиотеки flask. Сервис можно запустить на локальной машине в виде докер контейнера[14]. В таком случае наличие Python и его библиотек на локальной машине не потребуется. Пользоваться сервером можно через http запросы.

Кроме того, на языке Python разработана клиентская часть для сервера - *datapopclient*[14]. *Datapopclient* предоставляет удобный интерфейс для пользователей сервиса.

Сервис, подробную инструкцию по запуску сервиса и его использованию, а также клиентскую чать можно загрузить из репозитория *https://github.com/hushchyn-mikhail/DataPopularity*.

9 Доклады и публикации

Представленная в данной работе рекомендательная система презентовалась на устных докладах следующих конференций:

- 57-ая научная конференция МФТИ, "Оптимизация системы популярности файлов в экспериментах физики высоких энергий".
- 21st International Conference on Computing in High Energy and Nuclear Physics (CHEP2015), "Disk storage management for LHCb based on Data Popularity estimator".

Также результаты публиковались в сборниках соответствующих конференций.

10 Практическое применение

В настоящий момент рекомендательная система проходит тестирование в LHCb. Система и ее веб-сервис будут поддерживаться и улучшаться в дальнейшем с целью использования нашей рекомендательной системы для управления дисковой памятью системы хранения данных LHCb.

11 Заключение

В данной работе представлена рекомендательная система для управления дисковой памятью в гибридной системе хранения данных (жесткие диски и магнитные ленты). Было показано, что наш метод позволяет получить крайне низкое чило ошибочных удалений файлов с жесткого диска. Результаты показывают, что наша рекомендательная система позволяет добиться существенной экономии дискового пространства и значительно снизить среднее время доступа к данным.

Список литературы

[1] Hastie T., Tibshirani R., Friedman J. 2009 *The Elements of Statistical Learning* (Berlin: Springer)

[2] Hyndman R., Athanasopoulos G. *Forecasting: principles and practice* (https://www.otexts.org/book/fpp)

[3] Lipeng W., Zheng L., Qing C., Feiyi W., Sarp O., Bradley S. 2014 30^{th} *Symposium on Mass Storage Systems and Technologies (MSST): SSD-optimized workload placement with adaptive learning and classification in HPC environments* (California: IEEE)

[4] Beermann T., Stewart A., Maettig P. 2014 *The International Symposium on Grids and Clouds (ISGC) 2014: A Popularity-Based Prediction and Data Redistribution Tool for ATLAS Distributed Data Management* (PoS) p 4

[5] Beermann T. 2013 *Popularity Prediction Tool for ATLAS Distributed Data Management: J. of Phys.: Conf. Ser. 513 (2014) 042004* (IOP Publishing)

[6] Jamali S., Rangwala H., *Digging Digg: Comment Mining, Popularity Prediction, and Social Network Analysis* (http://cs.gmu.edu/ hrangwal/sites/default/files/GMU-CS-TR-2009-7.pdf)

[7] Quan H., Milicic A., Vucetic S., and Wu J. *A Connectivity-Based Popularity Prediction Approach for Social Networks* (http://www.dabi.temple.edu/ vucetic/documents/Quan12icc.pdf)

[8] Gupta M., Gao J., Zhai C., Han J. 2012 *Predicting Future Popularity Trend of Events in Microblogging Platforms* (https://www.asis.org/asist2012/proceedings/Submissions/207.pdf)

[9] Li J., Hong S., Xia S. 2012 *Neural Network Based Popularity Prediction For IPTV System* (J. of Networks, vol. 7, No. 12, Dec. 2012)

[10] Li H., Ma X., Wang F., Liu J., Xu K. 2013 *On Popularity Prediction of Videos Shared in Online Social Networks* (http://www.cs.sfu.ca/ jcliu/Papers/OnPopularityPrediction.pdf)

[11] Anonymous Author 2014 *Predict the Popularity of YouTube Videos Using Early View Data* (http://www.cs.ubc.ca/ nando/540-2013/projects/p16.pdf)

[12] Figueiredo F. 2013 *On the Prediction of Popularity of Trends and Hits for User Generated Videos* (http://homepages.dcc.ufmg.br/ flaviov/papers/figueiredo2013-wsdmdoc.pdf)

[13] *Python module and web service* URL https://github.com/hushchyn-mikhail/DataPopularity

[14] *Docker* URL https://www.docker.com

[15] *Reproducible Experiment Platform (REP)* URL https://github.com/yandex/rep