САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Математико-механический факультет

Кафедра Системного Программирования

Демьяненко Илья Игоревич

Распознавание приложений по запросам блочного уровня

Курсовая работа

Научный руководитель:
руководитель ИЛ RAIDIX Платонов С. М.

Санкт-Петербург
2014

Оглавление

Введение

Системы хранения данных используются в разных областях, в том числе в сфере работы с мультимедиа. Рынок мультимедиа (производство фильмов, телепередач) специфичен тем, что многие задачи имеют жёсткие сроки, в которые их необходимо выполнить. При этом с увеличением разрешения и количества видеоматериалов объёмы и число задач постоянно растут, а возможности СХД конечны, поэтому клиентам хочется распределять ресурсы в соответствии с важностью задач.

Компания RAIDIX предоставляет программное обеспечение для систем хранения данных, которое даёт возможности по управлению Quality of Service. Администратор может предоставить приоритет одному или нескольким инициаторам, тогда запросам от них будет предоставлена гарантированная пропускная способность.

Недостатком существующей реализации является необходимость управлять приоритетом вручную, что не всегда удобно. Задачи могут распределяться по инициаторам динамически, тогда придётся часто вносить изменения в список приоритетного обслуживания. Также существует дефицит грамотных администраторов, из-за чего компаниям приходится нести повышенные расходы на их содержание. Решение о том, предоставлять ли приоритет инициатору, принимается на основе задач, которые он выполняет в данный момент, поэтому, если знать, какие приложения в данный момент исполняются на клиентских компьютерах и генерируют нагрузку на СХД, можно принять решение о предоставлении приоритетов автоматически.

1. Исходные данные

Программное обеспечение RAIDIX предоставляет блочный доступ к RAID-массивам, поэтому в качестве исходных данных доступен поток запросов следующего вида:

- имя инициатора;
- время поступления запроса в микросекундах;
- тип запроса (чтение / запись);
- название RAID-массива;
- адрес;
- размер операции в секторах.

2. Постановка задачи

В рамках работы были рассмотрены следующие задачи:

- поиск закономерностей в поведении приложений. Поскольку поведение приложений не является случайным, потоки запросов от них, как правило, имеют характерные особенности, которые позволят отличить каждое приложение от других;

- изучение способов распознавания приложений с использованием найденных закономерностей; Эта задача является основной в данной работе. Необходимо исследовать особенности трафика приложений и на их основе разработать алгоритм(ы) их распознавания.

- разработка прототипа распознающего модуля и тестирование его эффективности. Разработанный алгоритм надо реализовать и интегрировать в существующее ПО системы хранения данных.

3. Выборочное распознавание

3.1. Описание

Поскольку задача ориентирована на мультимедиа-приложения, имеет смысл рассмотреть характерные особенности именно этого класса программ. Первой гипотезой было то, что приложения, работающие с мультимедиа, генерируют в основном последовательную нагрузку на СХД, что подтвердилось исследованиями бенчмарков, которые эмулируют работу видеоредакторов (Blackmagic Disk Speed Test, AJA System Test). Это позволяет ограничить класс распознаваемых приложений так, чтобы с ним было удобно работать.

Сузим круг до приложений, генерирующих линейную нагрузку и оперирующих блоками одинакового размера, которые, в случае бенчмарков, соответствуют кадрам видео. Условия значительно облегчают задачу, однако, после того, как запрос сгенерирован приложением, он проходит через стек ввода-вывода операционной системы и драйвер SCSI, в ходе чего может быть произвольным образом разбит на несколько запросов. Разрабатываемая система распознавания должна учитывать эти искажения.

3.2. Алгоритм работы

Я пришёл к системе, состоящей из двух компонентов: учителя и распознавателя. Учитель:

1. запускается в момент, когда работает только целевое приложение;
2. группирует запросы по последовательностям, которые соответствуют операциям линейного чтения / записи;
3. для каждой последовательности вычисляет размер исходных блоков и возможные варианты их разбиения транспортом;
4. если не удаётся найти размер блока, покрывающий 70% последовательности, считает, что последовательность нам не подходит;
5. если удалось найти размер исходного блока для 90% всех запросов, считает, что обучение удалось;
6. сохраняет информацию о размерах блоков и вариантах их разбиения для дальнейшего использования распознавателем.

Распознаватель:

1. обрабатывает поступающие на СХД запросы в потоковом режиме;
2. ведёт учёт последовательностей за последние N секунд;
3. каждый запрос либо продолжает известную последовательность, либо создаёт новую:
   (a) если запрос продолжает текущую последовательность, происходит её обновление;
   (b) в противном случае на его основе создаётся кандидат в последовательности. Кандидат становится полноценной последовательностью, когда он полностью соответствует нескольким блокам от одного из известных приложений; в этом случае распознаватель считает, что обнаружил новое приложение;
4. если какая-то из последовательностей не продолжалась в течение N секунд, она удаляется;
5. если у приложения не осталось работающих последовательностей, оно считается завершившимся.

3.3. Преимущества и недостатки

Преимущества:

- алгоритм является детерминированным (основан на чётких данных);
- приложения из целевого класса определяются безошибочно.

Недостатки:

- ориентирвоанность на узкий класс приложений;
- как выяснилось позже, реальные приложения для работы с видео ведут себя не так, как бенчмарки.

4. Вспомогательный клиентский модуль

Если мы хотим распознавать приложения со 100% достоверностью, можно задуматься о сборе дополнительной информации с клиента. Модуль, работающий на клиенте и перехватывающий системные вызовы приложений, имеет доступ к достаточному количеству информации для гибкого управления приоритетами. После сбора статистики по запросам приложений за небольшой промежуток времени модуль может отправить её на СХД либо по сети, либо с помощью собственных SCSI-команд. В этом случае СХД будет знать, какие приложения исполняются на СХД, какую нагрузку они генерируют, и с файлами какого типа они работают. Из файлов при этом могут извлекаться метаданные, что позволит различать даже работу с видео различного разрешения и автоматически определять необходимую пропускную способность по битрейту видео.

К преимуществам этого способа, помимо перечисленных выше, относятся низкие накладные расходы. На клиенте ведётся простой учёт операций, а на стороне СХД принимается решение по чётким правилам. Но также есть и недостатки: клиентскую часть придётся реализовывать под разные платформы (по меньшей мере, три), вносить изменения при некоторых обновлениях ОС, а также устанавливать дополнительное ПО на все клиентские компьютеры, что создаёт дополнительную работу для администраторов.

5. Машинное обучение

5.1. Описание

Машинное обучение используется во многих областях при решении задач любой степени сложности и имеет большие перспективы. Его суть заключается в том, что по тренировочному набору данных составляются вектора, хорошо характеризующие данные, затем на этом наборе векторов обучается модель. Далее по данным, которые нужно классифицировать, строятся вектора того же вида, которые распознаются моделью.

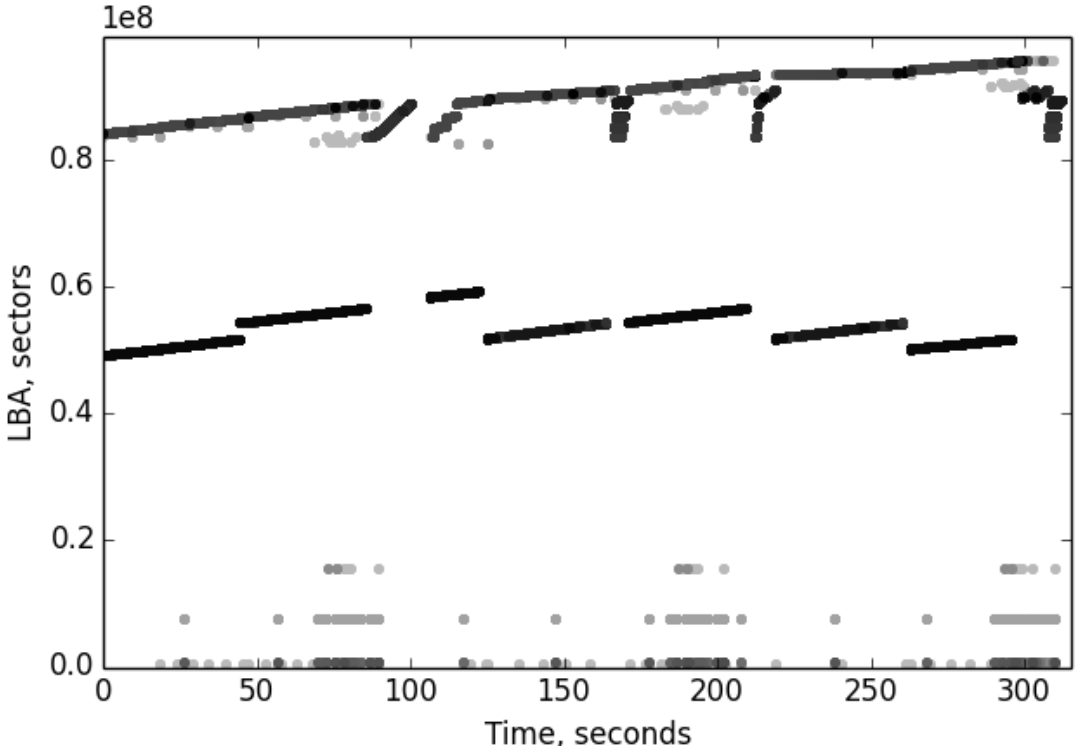

Рис. 1: Запросы к СХД от Apple Final Cut

После анализа разных трейсов с реальных мультимедиа-приложений я обнаружил, что большинство из них всё же производит операции последовательного чтения/записи. Это можно увидеть (рис. 1) на примере Apple Final Cut.

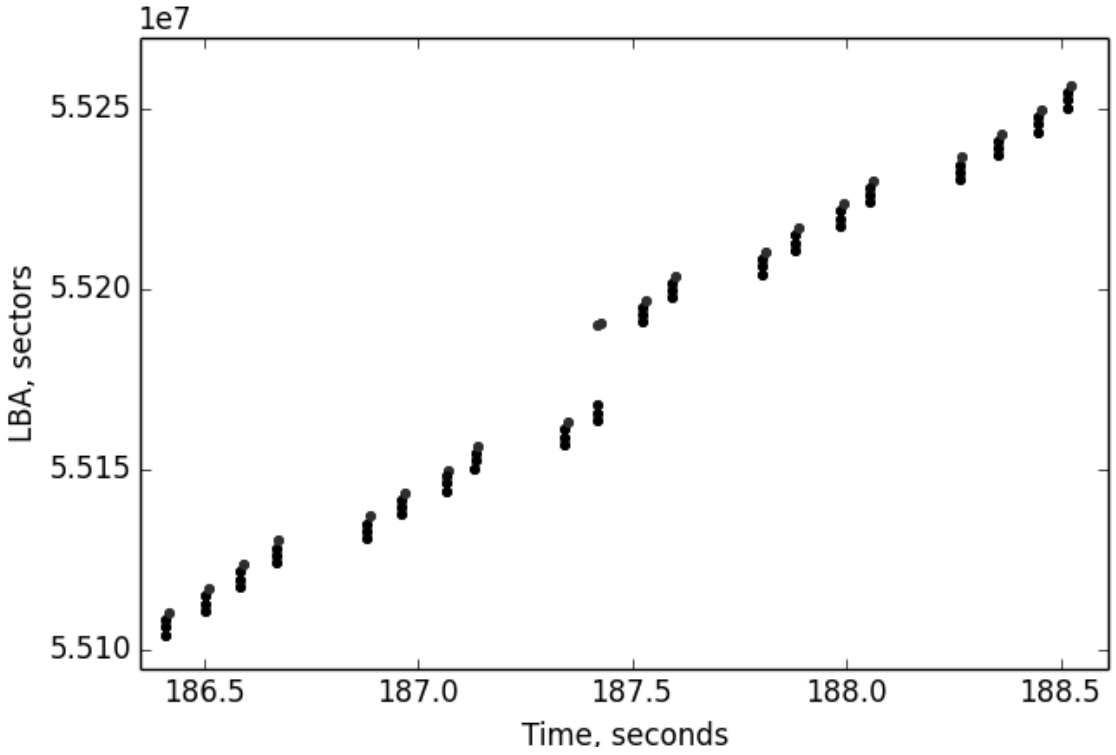

Рис. 2: Запросы к СХД от Apple Final Cut, уровень отдельных операций

При детальном рассмотрении оказывается, что запросы не совсем последовательны, то есть, конец предыдущего запроса не всегда совпадает с началом следующего (рис. 2). Тем не менее, если незначительно ослабить условие линейности, выясняется, что 90% запросов от реальных видеоредакторов относятся к последовательным операциям.

5.2. Алгоритм

В итоге я пришёл к следующему алгоритму обучения:

1. во множестве запросов производится поиск последовательностей по ослабленному условию;
2. каждая последовательность разбивается на интервалы по 20 секунд (целевое время отклика алгоритма);
3. на каждом интервале для каждого из 2048 различных размеров запросов собирается набор статистик:
   (a) размер запроса;
   (b) доля запросов этого размера среди всех запросов интервала;

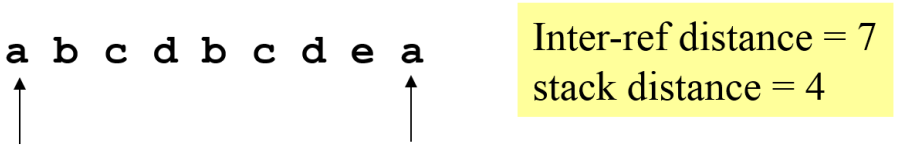

Рис. 3: Inter-reference distance, stack distance

   (c) mean inter-reference distance — среднее число запросов между соседними запросами выбранного размера (рис. 3);
   (d) mean reuse distance [1]. Эта статистика пришла из алгоритмов кэширования, где она равна числу обращений к уникальным адресам между двумя обращениями к конкретному адресу (рис. 3). В моём случае вместо адресов используются размеры запросов, полученный результат для каждого размера усредняется по интервалу;
4. набор статистик для каждого интервала объединяется в вектор.

Поскольку получилось много векторов, и многие вектора из соседних интервалов похожи, была предпринята попытка их кластеризации. Но из-за высокой размерности данных их не смогли кластеризовать за разумное время алгоритм k-means [2] и нейронная сеть Кохонена [3] из пакета WEKA. Также из-за неоднородности данных сложно выбрать метрику, которая позволила бы адекватно оценивать близость векторов.

Заключение

Было рассмотрено три подхода к решению задачи распознавания приложений.

Реализован модуль выборочного распознавания. Хотя он и ориентирован на узкого класса приложений, бенчмарки и операции резервного копирования определяются им безошибочно, что также имеет ценность для обеспечения QoS.

Подход со вспомогательным клиентским модулем рассмотрен с теоретической стороны, определены его достоинства и недостатки. Если признать разумной затрату ресурсов на реализацию и развёртывание клиентского модуля, этот способ будет самым надёжным, гибким и наименее требовательным к ресурсам СХД.

Машинное обучение является наиболее перспективным решением на стороне СХД. Возникшие проблемы могут быть преодолены путём уменьшения размерности векторов каким-либо способом, а также использованием алгоритмов, не нуждающихся в метриках.

Список литературы

[1] Evaluation Techniques for Storage Hierarchies / Mattson R., Gecsei J., Slutz D., Traiger I. // IBM Systems Journal. — 1970. — Vol. 9, no. 2. — P. 78–117.

[2] J. MacQueen. Some methods for classification and analysis of multivariate observations // Proc. Fifth Berkeley Symp. on Math. Statist. and Prob. — 1967. — Vol. 1. — P. 281–297.

[3] T. Kohonen. Self-organized formation of topologically correct feature maps // Biological Cybernetics. — 1982. — Vol. 43. — P. 59–69.