Доставка видеоконтента пользователям

Андрей Смирнов (NetStream)
РИТ: Высокие нагрузки (2008)

Введение

Что такое «контент» для видеохостинга? Во-первых, контент видеохостинга – это просто видео, которое представляет собой набор файлов в различных форматах, в частности, в формате FLV для просмотра пользователем через Flash Player. Эти файлы статичны, видеохостинг при загрузке пользователем видеоролика осуществляет конвертацию во все требуемые форматы с необходимым битрейтом. Хранение такого контента — это хранение обычных файлов, только довольно большого размера. Отдача контента — это, по сути, организация скачивания файлов.

Во-вторых, контент видеохостинга — это «живые» потоки или вещания. Вещания не записываются на диск, не происходит их конвертация, потоки раздаются клиентам с учетом пропускной способности каналов (происходит пропуск пакетов, если канал клиента недостаточен для получения потока вещания в полном качестве). Отдача контента в данной ситуации — это раздача потока на большое количество подключенных пользователей (тысячи смотрящих).

Кроме как таковой задачи отдачи контента (корректность, работа под нагрузкой и т.п.) актуальной является проблема «приближения» контента к пользователю, чтобы пользователь получал видео или вещание с сервера, который расположен близко к нему с точки зрения пропускной способности сетевых каналов, а также с точки зрения стоимости трафика для пользователя.

Этот доклад будет попыткой рассказать об одном из возможных подходов к решению поставленных задач. Так или почти так, как я буду рассказывать, работает видеохостинг Smotri.com, который является сегодня самым большим видеохостингом Рунета. Описываемые подходы могут быть применимы в областях с похожими характеристиками контента: достаточно большой объем файлов, много файлов, различные виды вещаний и т.п.

Содержание

Видео: организация файлового хранилища

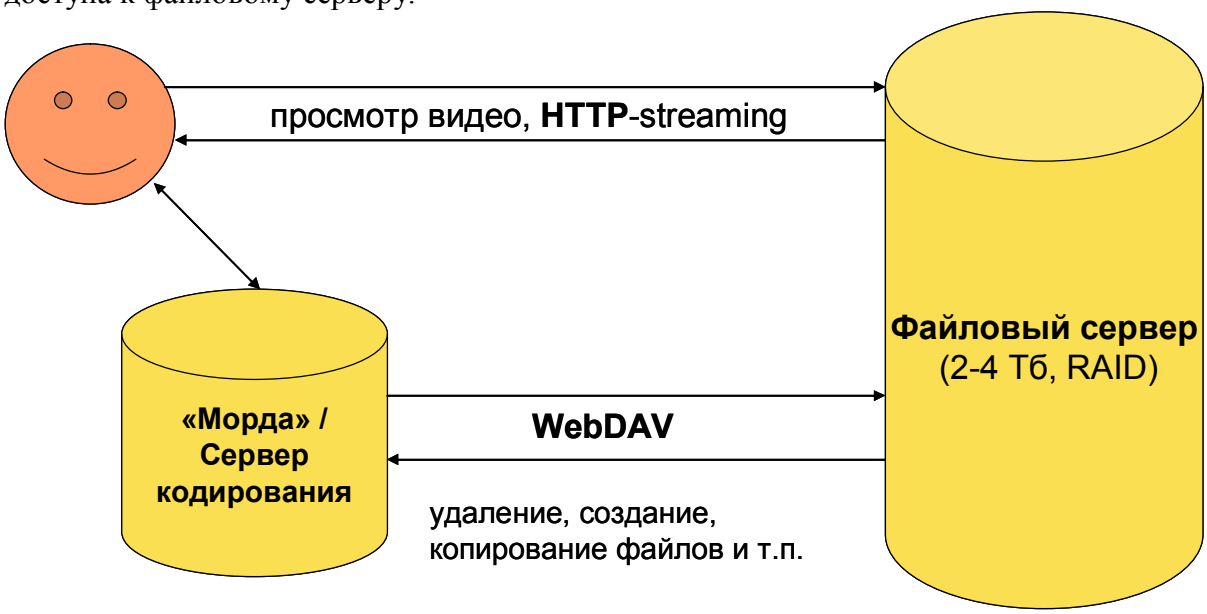
Видеофайлы

Видеохостинг в процессе своего функционирования получает видео в различных форматах, которое было загружено пользователем. Кроме хранения оригинального видеофайла, видеохостинг осуществляет конвертацию файла в ряд форматов: FLV, 3GP, MP4 и т.п. Все эти файлы необходимо хранить на файловых серверах, а также отдавать пользователям. Рассмотрим сначала вопрос хранения таких файлов.

Что представляют из себя эти файлы? Согласно нашим расчетам, в среднем для хранения одной секунды видео требуется 250 Кб дискового пространства (имеется в виду 250Кб/с на все форматы видео вместе взятые), при этом средняя длительность видео составляет примерно 4 минуты. Легко посчитать, что для хранения 1 млн. видеофайлов нам потребуется 60 Тб дискового пространства, что является уже достаточно внушительной цифрой. Этим объемом хранения надо эффективно управлять, а также обеспечивать раздачу такого объема контента пользователям. Кроме непосредственно видеофайлов нам придётся хранить и отдавать вырезанные из видео кадры, которых в нашем случае будет 15 штук: 5 кадров, вырезанные из разных участков видео, каждый из которых в трех различных размерах.

Доступ к файловому серверу через WebDAV

Непосредственно за хранение файла отвечает файловый сервер, в котором есть достаточно хорошо построенная дисковая подсистема в плане надежности и скорости доступа (RAID). Запросы на изменение файлов (создание новых файлов, удаление старых и т.п.) поступают с другой группы серверов: с «морд» (frontend) и с перекодировщиков. Морды – это обычные сервера, которые обслуживают подавляющее большинство запросов по HTTP как непосредственно к самому сайту, так и к его API. Перекодировщики принимают запросы от пользователей на загрузку исходных файлов видео и осуществляют их конвертацию. Итак, на морде или перекодировщике формируется понимание того, что нужно сделать с файлами на файловом сервере, т.к. файловая система не является локальной, необходимо то или иное сетевое средство доступа к файловому серверу.



Самое простое решение – это NFS, которое может сохранить для кода сайта «ощущение», что файлы находятся локально, забирая при этом на себя все сложности по выполнению удаленных операций через сеть, однако NFS может вести себя крайне ненадежно при падении соединения между мордой и файловым сервером, а также в силу

того, что и морд, и файловых серверов десятки, количество кросс-маунтов NFS становится слишком большим.

Самым простым решением в такой ситуации может быть WebDAV. WebDAV является расширением HTTP, который, кстати, изначально предполагал, что содержимое World Wide Web является не статичным, а изменяемым. WebDAV делает еще один шаг вперед, добавляя в HTTP полноценные возможности по удаленному изменению, созданию, удалению файлов, каталогов, получению информации о файлах и т.п.

Внутри кластера серверов видеохостинга обращение к файловым серверам идёт по протоколу WebDAV. Скачивание файлов (в том числе проигрывание видео в Flash Player) идёт напрямую с файлового сервера, что позволяет более равномерно загрузить сетевые каналы.

Выбор файлового сервера из кластера

Предположим, у нас есть кластер файловых серверов, все настроены, готовы отдавать файлы, а также принимать новые файлы через WebDAV. Было загружено видео. Какой файловый сервер выбрать для хранения очередного файла? Мы пробовали несколько стратегий выбора, можно предложить следующие характеристики для выбора:

- объем свободного места/объем сервера;
- нагрузка на сервер (как измерять?);
- случайный выбор.

Примером плохой стратегии является стремление заполнить все сервера так, чтобы процент занятого места на сервере (или просто объем свободного места) был равным среди всех серверов кластера. При таком подходе обеспечивается равномерное заполнение серверов, однако на практике проект начинается с нескольких файловых серверов, затем добавляются еще и еще, и получается, что при добавлении нового сервера в кластер все новые файлы складываются именно на этот сервер. А новые файлы чаще всего оказываются и самыми популярными в данный временной отрезок, поэтому нагрузка на новые файловые сервера в такой конфигурации многократно возрастает.

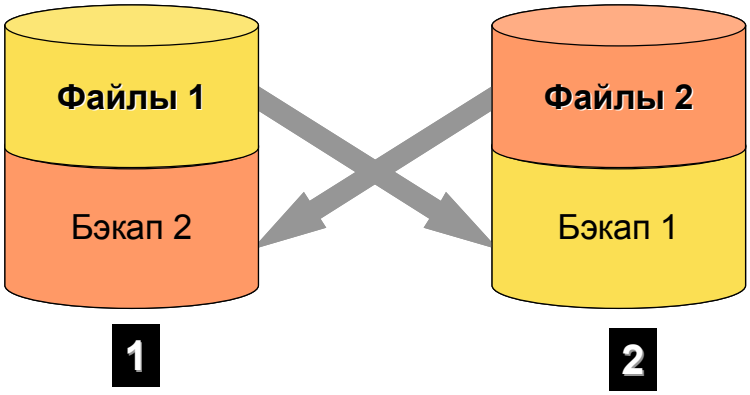
Гораздо более удачной является такая стратегия: среди всех файловых серверов выбираются те, уровень заполнения которых не достиг некоторой критической отметки (т.е. те сервера, на которых еще есть свободное место), а затем среди них осуществляется случайный выбор (возможно, выбор с учетом веса сервера). При такой стратегии популярные видео оказываются равномерно распределенными среди большего числа серверов, что позволяет более равномерно распределить нагрузку по скачиванию контента.

Необходимое ПО

Итак, описанный подход требует простого и распространенного программного обеспечения: для скачивания файлов подойдет любой «быстрый» HTTP-сервер, например, nginx, lighttpd и т.п. При этом для обеспечения подгрузки FLV в плеер с любой временнóй отметки (так называемого «стриминга»), нам необходимо очень простое расширение, которое сегодня присутствует как в nginx, так и в lighttpd. В качестве WebDAV-сервера подойдет Apache httpd и т.п. Для доступа к файлам по WebDAV можно использовать любой WebDAV-клиент, который есть в том или ином виде практически во всех языках программирования. Так, например, в PHP он реализован в виде PEARовского класса (который приходится «допиливать», чтобы обеспечить разумную производительность). WebDAV не обеспечивает функциональности по получению объема свободного места на диске, но это легко обойти с помощью простейшего скрипта в cron, который вывод df перенаправляет в файл в корне файлового сервера, а этот файл уже можно скачать по HTTP с любой морды.

Кросс-бэкап

Каким образом обеспечить надежное сохранение 60 Тб данных? (При условии, что эта цифра будет расти?) Традиционные варианты с бэкапом на центральный сервер уже не подходят, т.к. такой объем хранения на одном сервере обеспечить трудно, да и надежность такой схемы невысока. Никакой уровень RAID в пределах одного сервера не может гарантировать надежность, т.к. возможен и аппаратный, и программный сбой, приводящий к потере данных. Простым и достаточно надежным способом является кросс-



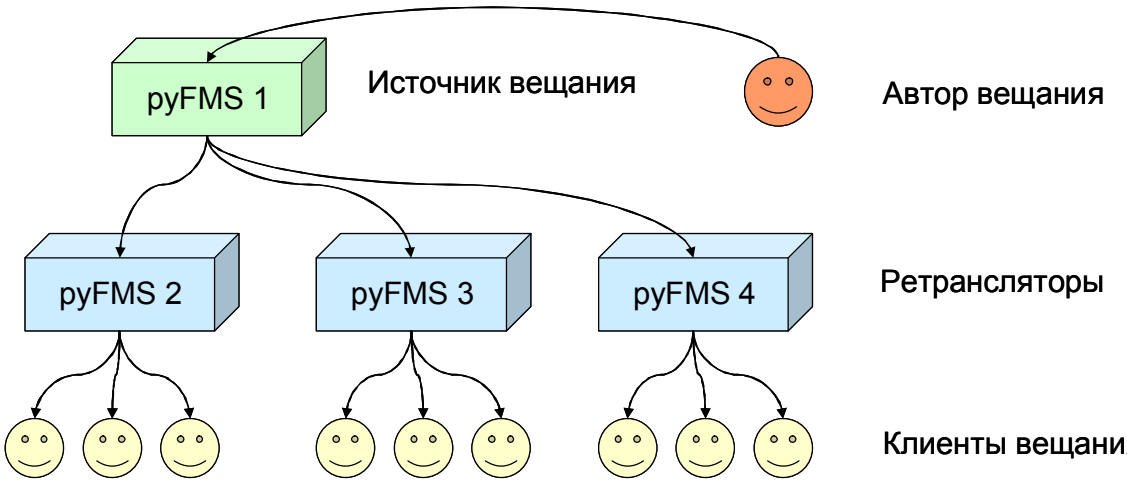
бэкап: каждый из файловых серверов заполняется наполовину, а затем содержимое первого сервера бэкапится на второй, и наоборот. Таким образом, если произойдет сбой на одном из серверов, на втором останется полное содержимое обоих серверов.

Для реализации данного подхода необходим простой управляющий модуль, а также rsync. При использовании кросс-бэкапа интересным является то, что по одной информации о свободном и занятом пространстве на файловом сервере невозможно судить о его заполненности, т.к. после выполнения синхронизации бэкап-части возможно произвольное увеличение объема хранения (если заранее неизвестна разбивка: сколько места занимают сами данные, а сколько занимает бэкап другого сервера).

Вещания: ретрансляция

Подробный рассказ про организацию вещаний и нашем сервере вещаний был уже представлен на конференции РИТ-2008, поэтому здесь я не буду повторяться и вдаваться в подробности, остановлюсь лишь на основных аспектах.

Вещание представляет из себя видео и аудио потоки, которые кодируются на стороне клиента с помощью Flash Player, затем с помощью протокола RTMP отправляются на сервер вещаний, который раздает эти RTMP-потоки всем подключенным к вещанию зрителям. Здесь основной проблемой является то, что поток раздается клиентам хоть и без перекодирования (в качестве и с битрейтом, заданным автором трансляции), но количество клиентов, подключенных к вещанию, может составлять тысячи человек. При этом каждому клиенту может доставляться измененный поток в соответствие с пропускной способностью каждого клиента (пропуск пакетов в потоке). Кроме того, в RTMP-потоке кроме собственно потока вещания происходят удаленные вызовы процедур (сервер-клиент или клиент-сервер), а также транслируется состояние разделяемых объектов между всеми подключенными к вещанию клиентами.



Один сервер не всегда может справиться с задачей раздачи потока вещания, поэтому мы прибегаем к ретрансляции: автор вещания подключается к первичному серверу вещаний, к которому в качестве клиентов присоединяются другие сервера вещаний, которые уже в свою очередь раздают поток конечным зрителям вещания. При такой схеме оказывается возможным равномерно распределить нагрузку между серверами вещаний (количество клиентов на каждом сервере оказывается примерно одинаковым).

Географическая распределенность: подход

Необходимость географической распределенности

Кроме самого факта, что контент был доставлен пользователю, мы должны обеспечить *качество* доставки контента. Для FLV-файла видео это означает, что скорость, с которой он доставляется пользователю, должна быть выше либо равна битрейту файла, иначе видео у пользователя при просмотре будет «затыкаться».

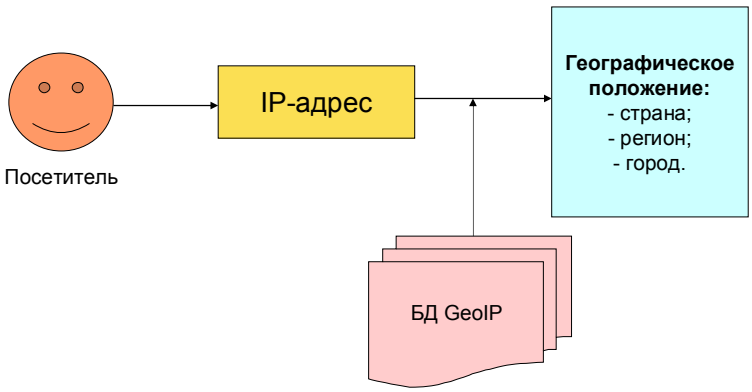
Кроме того, имеет смысл «приблизить» контент к пользователю географически. Это связано с пропускной способностью каналов (отсутствием иногда хороших магистральных каналов), а также с разницей в стоимости локального и внешнего трафика для конечного пользователя (например, в регионах РФ).

Такой шаг необходимо сделать при желании выйти на международный рынок, а также при региональном развитии внутри РФ. Сегодня в регионах очень часто самыми популярными сайтами являются региональные порталы, которые предоставляют различные сервисы, в том числе и сервис видеохостинга, а их популярность обусловлена как стоимостью трафика, так и скоростью доступа/временем отклика. Можно представить, что пользователь готов подождать открытия страница, загрузки плеера, но тяжело предположить, что пользователь согласится смотреть видео, которые прерывается из-за постоянной буферизации, или смотреть вещание, которое доходит до пользователя в виде слайдшоу (после пропуска пакетов остались только опорные кадры видео).

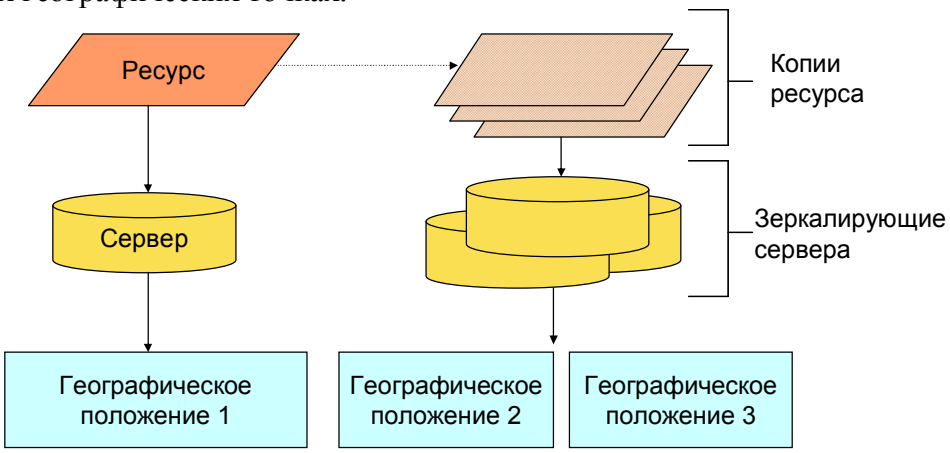
Таким образом, осознав необходимость географической распределенности для контента, мы покупаем/арендуем сервера в непосредственной близости от потребителя: в Европе, США, Украине, Екатеринбурге и т.д. Что же делать дальше?

Механизм работы

В абстрактном варианте в процессе доставки контента участвуют две сущности: ресурс, наш контент (например, вещание или видео), и посетитель. Необходимо для начала узнать, где находится наш потребитель ресурса – посетитель нашего сайта. Надеяться, что он сам расскажет эту информацию о себе, не приходится, поэтому мы берём IP-адрес посетителя, выполняем поиск в базе данных GeoIP (коих сегодня довольно много, как платных, так и бесплатных), и получаем на выходе информацию о местоположении пользователя: его страну, регион, город, название его провайдера.

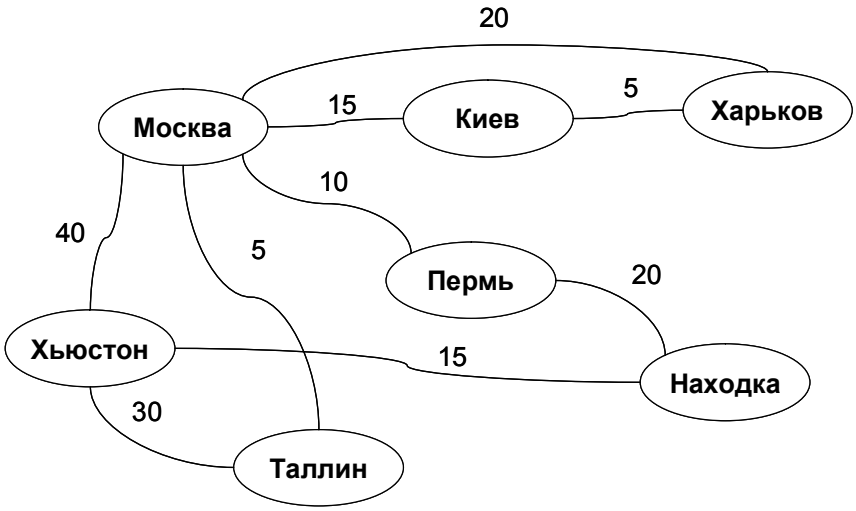


Ресурс (контент) расположен всегда на конкретном сервере, а сервер имеет своё физическое местоположение, заранее нам известное. Поэтому ресурс через сервер тоже обладает своим географическим местоположением: те же страна, регион, город и т.п. Кроме того, мы можем создавать копии ресурса на специальных зеркалирующих серверах, таким образом один и тот же ресурс будет доступен на нескольких серверах, а значит, в нескольких географических точках.



Выбор ближайшего к посетителю ресурса

Теперь у нас есть посетитель, его местоположение, ресурс, который он хочет получить вместе со всеми его копиями и местами их расположения. Теперь необходимо выбрать именно тот ресурс, который ближе всего к пользователю. Как это можно сделать? Логично было бы посчитать расстояние от посетителя до ресурса и всех его копий и выбрать самый близкую к посетителю копию ресурса. Каким образом задать такое расстояние?



Это расстояние не всегда совпадает с расстоянием на карте между двумя точками, и является скорее мерой качества и пропускной способности каналов между регионами,

странами или городами (или даже между отдельными провайдерами). Мы можем считать, что если регион входит в какую-то страну, то расстояние между ними мало, аналогично для городов и регионов. В первом приближении достаточно задать расстояние между отдельными странами.

Таким образом, мы получаем взвешенный ориентированный граф, на котором надо решать задачу поиска кратчайшего пути (минимизация суммы весов ребер графа, входящих в этот путь). Это классическая задача, которая может быть легко решена за полиномиальное время. Граф хоть и является слабосвязанным, но всё-таки его размеры достаточно велики, поэтому в реальном времени для каждого посетителя выполнять такой расчет не является самым эффективным решением. Но мы можем немного «схитрить»: мы знаем, что при всех поисках кратчайшего пути конечным пунктом пути будут места, где расположены сервера с ресурсами, таким образом число концов пути фиксировано и относительно невелико. Теперь нам достаточно рассчитать и закэшировать, например, в memcached, длины кратчайших путей от всех вершин графа (где могут располагаться посетители нашего сайта) до мест расположения ресурсов.

Уже эта информация будет использована в реальном времени, когда за ресурсом обратится посетитель. Если мы найдем несколько копий ресурса, на которых будет достигаться кратчайшее расстояние от посетителя, мы выберем любую из копий случайным образом (возможно, в соответствие с весом того сервера, где расположен ресурс).

Копирование ресурса

Схема выбора работает замечательно, когда мы имеем копии ресурса на серверах, разбросанных по всему миру. Однако как эти копии создаются? Было бы неразумно копировать все ресурсы на все сервера, лучше выбирать именно те ресурсы, которые будут востребованы конкретной аудиторией.

Для этого мы предлагаем следующее решение: когда посетитель обращается за ресурсом, всем географическим местам, где мог бы находиться ресурс, но в данный момент не находится, начисляется бонус:

$$бонус = \frac{k}{расстояние}$$

где *расстояние* – это расстояние от посетителя до данного географического места, а k – некоторый коэффициент, который задает то, насколько быстро ресурс будет перемещен на указанный сервер. Если нет пути от посетителя до данного места (т.е. посетитель далеко), то расстояние будет равно бесконечности, а бонус будет равен нулю. Если же путь существует, то ресурс будет скопирован на сервер в данной точке тем быстрее, чем ближе он к посетителю и чем больше таких посетителей попросят доступ к данному ресурсу.

Как только бонус превышает некоторый порог, осуществляется копирование ресурса на сервер, расположенный в данном географическом месте, и в дело уже вступает алгоритм выбора копии ресурса, описанный выше.

Географически распределенные видеофайлы и вещания

Теперь мы можем применить описанный выше подход к контенту видеохостинга: вещаниям и видео.

Видеофайлы. В этом случае ресурс – это сам файл видео (причём любого типа, как FLV, так, например, и оригинал видео). Копии ресурса – копии файла, находящиеся на зеркалирующих файловых серверов. Обращение к ресурсу – скачивание файла, проигрывание FLV-видео посетителем в Flash Player'е. Копирование ресурса – это просто копирование файла с основного файлового сервера на один из зеркалирующих файловых серверах, расположенных в различных точках.

Вещания. Для вещаний ситуация очень похожа, здесь ресурсом является, конечно же само вещание, расположенное на основном для него сервере вещания (том сервере, куда подключен автор вещания). Копии ресурса – это все ретрансляции данного вещания на других серверах вещания. Обращение к ресурсу – это «вход» в вещание нового посетителя. Копирование ресурса – открытие ретрансляции на сервере вещаний, расположенном в той или иной точке мира.

В случае вещаний интересным является то, что ретрансляция является одновременно способом борьбы с нагрузкой и средством приближения контента к потребителю, то есть ретрансляция осуществляется одновременно в двух плоскостях: с одной стороны, удовлетворить всех пользователей наиболее близким контентом, с другой стороны необходимо обеспечить в каждой географической точке такое количество ретрансляций, чтобы нагрузка на сервера вещаний оставалась в норме.