Review - 12 Spotify – Large Scale, Low Latency, P2P Music-on-Demand Streaming

When creating a large scale music-on-demand streaming service, low latency is one of the core requirements. In 2010, Spotify had a user base of over 7 million users in 6 countries. By combining streaming from main servers with P2P streaming across clients, Spotify significantly reduced the load on their servers while keeping the low latency needed for seamless streaming.

Users of Spotify is able to play tracks continuously, skips tracks and seek within a track in the client. Those actions should feel instant to the user, and minimize the time required to buffer the next track to be played.

The current paper presents a quantitative analysis of how the client chooses between peers and the Spotify servers, while also maintaining a cache of the recently played tracks to reduce the latency.

The paper analyses data collected from both clients and servers over a 7 days period and looks into data on the number of tracks played, the latency for each track, stutter and how the client chooses which peers to retrieve a track from while streaming.

The study reveals that roughly 56% of the tracks were streamed from peers, almost 34% from cache and the rest from Spotify Servers. Most users, about 62% experienced between 100ms and 500ms latency, 265ms on average. 5% were measured to be over 10s, but this is explained to be erroneous measurements.

While streaming from the Spotify servers, tracks are served in 10s chunks, allowing the client to further stream the rest of the track from other peers if it is available.

To keep the network overhead low, the Spotify client keeps a small list of other peers it can connect to. When a client wants to download a track, it sends out a notification to its peers containing the name of the track and a priority, e.g if the track is currently streaming the track, prefetching the track or syncing it for offline synchronization.

The overhead in the P2P network is measured to be roughly 5.2%, measured as \textit{not useful song data}. 2.38% were song data downloaded not useful to the client and the remainder were search data and other overhead. As the paper discuss, this number is slightly higher than what is measured in BitTorrent networks, but the protocols are quite different, and the data gathering methods varies.

The paper refers to further work in the field, such as the development of user satisfaction for on-demand streaming, improved playout strategies for P2P delivery and efficient P2P overlays exploiting the overlap in interests between users.

Review - 12 Spotify – The Hidden Pub/Sub of Spotify (Industry Article)

Spotify is not only a music streaming service, it is also a social hub, providing a live and offline feed of updates to its users. To enable this feed, Spotify uses a hybrid pub/sub system providing both real-time and offline notifications to its users.

The notification system consists of multiple generators and subscription models. Users can befriend each other, subscribe to playlists and follow artists. This results in the friend feed, playlist updates and notifications related to artists. The friend feed consists of real-time data, which is pushed directly to the subscribers. Playlist updates and notifications about artists are also pushed to the live feed, but is persisted to secondary memory as well to ensure all subscribers get at least one notification on all of their devices.

The paper analyses the communication between subscribers, publishers, the notification module and the access points.

Real time data is handled by the pub/sub engine, which is a lightweight engine publishing notifications to online subscribers. As discussed in the paper, the data published through the pub/sub engine is of a massive amount and not suited for storage. The pub/sub engine also publishes updates on a best-effort delivery, and does not require any acknowledgement from the clients.

The analysis of the subscriber data resulted in some interesting findings. Topic popularity and subscription size follows a power law distribution, but when studying the relationship between the number of subscribers to a
topic and the correspondent publication rate, there was no correlation at all. Meaning that a topic with few subscribers could have many more publications than a topic with many subscribers.

The paper also describes the distribution of publication traffic, where most of the presence traffic is generated on weekdays, with peak traffic at 6PM and the lowest traffic at 2AM. Playlist traffic follows the same pattern, except for the distribution being fairly even throughout the week. Notification traffic, including artist updates and social services results in small spikes every day, being the result of those notifications being generated by batch jobs.

The paper leaves some open questions for further research, for instance the low ratio of subscribers to users listening a lot to music and vice versa.

**Spotify – Large Scale, Low Latency, P2P Music-on-Demand Streaming**

Spotify is a music streaming service offering low-latency access to a library of a large music tracks. Spotify adopted TCP as a transport protocol. Combination of server-based and peer-to-peer streaming gives a good performance. Considering two situations in behavior of users, sometimes they access track in order, sometimes in random. Combining server and peer-to-peer streaming and using cache decrease the load on Spotify’s servers and bandwidth efficiently, furthermore this approach makes latency lower and stutter lesser. In this article, it also shows the analysis of the performance which are obtained through several measurement of the system.

**The Hidden Pub/Sub of Spotify (Industry Article)**

This article give us a detailed expression of the architecture of Spotify and studies the design for pub/sub system. At last, it collect data from the production system and give some analysis of pub/sub system. There are 4 kinds of service and 3 kinds of notification in this architecture. At the end it gives some result about the relationships between topic popularity and subscription size, relationships between publication traffic subscription traffic and time and so on.

**Spotify - music streaming and related social interaction**

These two papers are describing the social network of Spotify - a music streaming application which is using the aspects of social interaction in two major ways. The first (and basically the milestone of Spotify) is the use of peer-to-peer network for sharing music among the users. The second is the social connection which can be seen on the outsideand that is sharing your music music appetite (either by direct subscription or by a Facebook friendship).

The first paper basically sums up how the actual receiving of a track works. The main idea of Spotify is to use peer-to-peer network as much as possible, so that the Spotify servers won’t be that overwhelmed by traffic. It describes (a) the system of caching (prefetching a next song), that is when to start the prefetching, how are the cached tracks used, etc., (b) the system of finding the right peer to download a track from and (c) the maintainance of peers.

On the other hand, the second papers focuses mostly on the problem of subscriptions to topics rather then to the way how the actual files are served. It shortly describes how Spotify is using the pub/sub paradigm, that is (a) what topics a user can subscribe to and (b) which features can produce an event for subscribers. It also describes how the architecture for the paradigm is designed - (a) how are the events handled (whether they are sent immediately or stored) and (b) the services which can produce events. It thouroughly describes single modules of the system and describes their purpose.

Both papers evaluate different measurements of Spotify. The first one focuses mainly on the contributions of subscriptions to topics rather then to the way how the actual files are served. It shows that even though less then 9% of tracks were downloaded from the servers (and the rest from peers), playback latency was in an acceptable range and shutter didn’t appear in more then 1%.

The evaluation content of the second paper studies the characteristics concerning topics and the system of subscriptions. It shows that certain characteristics follow the power law (Topic Popularity) and some don’t (Publication Event Ratew). It also shows the division of publication traffic among different event sources as well as division of different subscription types. From boths it can be seen that the major source of Spotify activity is the friendship feature.
There are still many aspects of the Spotify service that could be monitored, analyzed or further developed - for example the access pattern of music or more efficient P2P overlays (using the overlap of people’s interests). The use of P2P can significantly reduce the traffic over servers without violating the flow of the service. The friendship aspect of Spotify is the leading force of the social aspect of the site.

**Spotify – music streaming and social interactions**

Spotify is a very popular platform for streaming music, it is steadily growing since it was launched in 2008. The service offers over 20 million tracks and currently has over 75 millions active users worldwide. As with any streaming service the main concerns are playback latency and stutter, which have to be minimised in order to give user the best performance. Furthermore Spotify is providing users with tools for social interactions within the service. Users are able to connect with other users and subscribe to certain playlists or artists.

Clearly, creating and managing such a huge service is no easy task, and can be approached in many ways, some of which may prove to be impossible to manage as the service grows. Fortunately Spotify engineers have implemented a service in way which allows for usage smooth enough that the user base is growing. The question is: How they Gaining access to Spotify architecture details as well as usage data is a great source of knowledge about ways to implement such a huge streaming service.

The papers present an overview of some of the architecture and protocols used by spotify, in addition to analysing usage patterns.

First paper “Spotify – Large Scale, Low Latency, P2P Music-on-Demand Streaming” describes ways which allow Spotify to minimise latency and stutter. These ways are using a TCP connection to download music, prefetching and caching music and most importantly usage of peer to peer network as mean of decreasing servers load.

Second paper “The Hidden Pub/Sub of Spotify” presents the architecture details and overview of the social aspect of the service. That aspect is implemented using a pub/sub paradigm. Furthermore paper describes how service deals with non-local subscription requests.

Both papers also provide some overview of the usage statistics of the service, the main conclusion is that the service is used mostly during the day - during weekends as well as weekdays.

Papers provide a good overview of the Spotify architecture, but it should be noted that the p2p aspect of the spotify clients was removed because of growth in the service native infrastructure.

**Spotify – Large Scale, Low Latency, P2P Music-on-Demand Streaming**

The paper is an evaluation of Spotify, which is a large scale music streaming service.

The paper gives an overview of the protocol and peer-to-peer architecture used and provides measurements of service performance and user behavior. What makes the study worthy is the scale of the users Spotify serves. Spotify currently has a user base of over 7 million. Another important feature of Spotify, which is measured and highlighted in the paper, is the service low latency.

The paper shows that the architecture employed by Spotify, which is a combination of client-server and peer-to-peer, results in good quality for a music streaming service. The paper also makes a discussion about user access patterns and how the peer-to-peer network affects the access patterns as they reach the server.

I certainly believe that the authors had a good motivation for doing this study. Large scale streaming service with low latency is something of interest to multimedia research community. I also think that the paper covers the subject very good. All the aspects from peer-to-peer architecture till user access pattern are comprehensively discussed as they were mentioned in the abstract.

**The Hidden Pub/Sub of Spotify (Industry Article)**

This paper is about a world-known music streaming service called Spotify. The paper gives an analysis of a hybrid pub/sub system designed for real-time as well as offline notifications for Spotify. The pub/sub system provides Spotify users with several social interaction features.
The authors explain how the architecture allows the users to follow playlists, artists, and the music activities of their friends. They provide detailed explanation about the end-to-end architecture of the pub/sub engine at Spotify. They also examine system’s performance using recorded traces. In their performance evaluation, they focus on different features such as the workload of the pub/sub system and the message traffic produced by the pub/sub system.

I believe that the motivation for doing this study is highlighted very well. In addition, the authors gives a through explanation about how the pub/sub Spotify system works.

**Spotify—Large scale, low latency, P2P music-on-demand streaming**

The traditional client-server on-demand streaming service cannot support too many users due to the bandwidth and resources limit. If the number of users exceeds a given threshold, high latency becomes unavoidable. Clearly, providing a large scale, low latency on demand streaming service is desirable. P2P network has been well researched in the past for breaking bandwidth and resource limit. It utilizes user resources and bandwidth to relieve server stress. The question is: how to combine P2P network to provide large scale and low latency music-on-demand streaming?

This paper presents a real instance Spotify and its peer-to-peer architecture, protocols and evaluation.

The paper discusses the general streaming protocol and extend it to Spotify’s peer-to-peer network. It examines the performance of different strategies under an extensive set of real-world traces.

The result is quite attractive: during the measurement period, only 8.8% of data came from server and less than 1% of playback latency and less than 1% of all playbacks stuttered.

It is better to give a description about the relationship between different parts.

The paper shows that a proper global overview is important for readers to understand better.

**The hidden pub/sub of Spotify**

The pub/sub system is a distributed system where subscribers get messages from publishers. There are many existing applications, such as Facebook, Spotify and Youtube. However, there are few works trying to explain the pub/sub system used in those well-known websites and applications.

Therefore, providing a case study is desirable for those who want to know the pub/sub system.

This paper presents the design of the hybrid pub/sub system used in Spotify and its performance under the real application.

The paper first introduces the Spotify and then presents the model and features of the Spotify pub/sub system. After that, the architecture is illustrated and the problem about real time service and persisted service to online clients is discussed. The notification module and pub/sub engine is detailed before the performance analysis.

The results from performance analysis are very interesting. First, topic popularity and subscription sizes follow a distribution close to a power law while publication event rate does not follow power law. Second, subscription cardinality of a subscriber is very low and linearly proportional number of topics subscribed by that subscriber. Third, the publication event rate of a topic does not rely on its popularity. Fourth, publication traffic follows a daily pattern. Fifth, subscription and unsubscription rates significant churn in subscriptions.

This paper gives a case study of a pub/sub system, which is helpful for people who want to learn this kind of system. The results from performance analysis are attractive and can be used to guide some related applications.

It is common to use figures to illustrate the model. This paper shows that it is more readable for readers.

**Summary of Spotify—Large scale, Low Latency, P2P Music-on-demand Streaming**

This paper performs an in-depth study and evaluation of Spotify. Spotify streams over 8 million tracks to more than 7 million users in Europe. It has a very low playback latency which is 265 ms. Client/Server and peer to peer technology are used to distribute music. A Spotify server is accessed when a chunk of music is in need urgently, less urgent music chunks are downloaded from the peer to peer network. Caching and prefetching are
used to reduce latency. Music random access is realized by requesting the first 15 seconds of a randomly accessed music from a server and then downloading the rest of the music from peers.

The rationale behind using a peer to peer system is to increase scalability by offloading Spotify servers. The P2P overlay used is a tracker based unstructured network. A client looks for peers that have the required music in two ways simultaneously. On the one hand, it requests a list of peers from a tracker. On the other hand, it broadcasts a query to its one hop and two hop neighbors. A client is configured to only connect to a certain number of peers at a time. If this number is greater than the hard limit, some peers need to be disconnected. Spotify clients use 6 criteria to sort and select peers.

Spotify was evaluated using Munim databases and log files. The authors observed that the usage was high during weekdays. Based on the authors, 8.8% of data came from servers, 35.8% from the p2p network, and 55.4% from local cache. The observed median latency was 256 ms. Less than 1% of the playbacks stuttered during the measurement period. 79% of users listened to 21% most popular music on Spotify. 5.2% of received data was considered as cost.

Summary of the hidden pub/sub of Spotify

This paper explains and evaluates the hidden pub/sub system of Spotify. Spotify employs a three-layer pub/sub architecture. The bottom layer comprises event brokers that form a DHT. The second layer consists of aggregators that aggregate subscriptions for the same topic. The third layer includes access points that interact directly with Spotify clients. A Spotify client sends its subscriptions to an access point, which forwards the subscription to an aggregator. A client can subscribe to updates from friends, playlists, and artists. Subscriptions to the same topic from different subscribers are aggregated as one subscription by the aggregator to reduce the number of subscriptions stored by the event brokers, hence, make the system scalable. An aggregator chooses an event broker based on the hashed topics. The subscriptions are stored in the memories of event brokers. An event broker at one site has a mirror broker at another site. They store the same contents. There are three types of publication services — best effort real-time delivery, persisted delivery to online users, and persisted delivery to offline users. User presence is delivered by best-effort real-time delivery supported by the pub/sub engine which comprises event brokers. Other publications are delivered by the notification module, which consists of a rule engine, a notification service, and a Cassandra cluster. The updates of playlists, social service, and artists are stored into Cassandra’s events column. The rule engine polls the events column for new updates and decides if notifies users immediately or stores them into the notifications column for later delivery. When an offline user gets online, the notification service queries the notifications and timestamps columns to retrieve the new notifications since the last event received by the client.

The authors analyzed the pub/sub system and got the following observations. Topic popularity and subscriptions sizes follow a distribution close to a power law. The publication event rate doesn’t follow power law. Subscription cardinality is significantly low and varies. Publication traffic shows a daily pattern. Publication traffic from local sites is much higher compared to that from remote sites. Subscription and un-subscription cause churn.