May 03, 2018

Performant Shuffling Heuristic for Arbitrarily Large Playlists

Senthil Palanisami

PERFORMANT SHUFFLING HEURISTIC FOR ARBITRARILY LARGE PLAYLISTS

By: Senthil Palanisami

Audio and video content can be stored on data servers and provided to users for listening/viewing over the Internet. A content sharing platform may allow users to upload, view, and share content, such as video content, image content, audio content, text content, and so on (which may be collectively referred to as "media content items"). Such media content items may include audio clips, movie clips, TV clips, and music videos, as well as amateur content such as video blogging, short original videos, pictures, photos, other multimedia content, etc. Applications for supporting the listening/viewing of such media content items may be browser-based, or may run independently of a browser in a standalone application.

In some instances, a content sharing platform may provide a playlist of the media content items. The playlist includes certain media content items to play for the user based on a selection by the user (e.g., a channel related to a genre, band, artist, era, etc.) and/or the preferences of the user (e.g., the media content items "liked" by the user, a number of times the user has played certain media content items, etc.), among other things. The content sharing platform may create the playlist and arrange the selected media content items in the playlist.

A user may select an option in a user interface of the content sharing platform to shuffle the media content items in the playlist to watch and/or listen to media content items in the playlist in a random order. Upon selection of the shuffle option, the content sharing platform may randomize the order of playback of the media content items in the playlist. Conventionally, the randomized playlist is not presented to the user in the user interface. That is, conventionally, the user may not see what media content items in the randomized playlist are going to play after the media content item that is currently playing. Such a presentation may enable a user to skip any

media content items that are to subsequently play in the randomized playlist to which the user is not interested in viewing and/or listening.

Further, some conventional systems may suffer from performance issues when randomizing playlists that include an arbitrarily large (e.g., hundreds, thousands, tens of thousands, etc.) number of media content items. For example, in some conventional systems, a large (e.g., ten thousand) playlist of videos may be obtained from a database and the order of the ten thousand videos in the entire playlist may be randomized, which may incur significant performance costs and take an unreasonable amount of time to complete. This may provide an undesirable user experience due to the delay in performing the shuffling.

In some instances, instead of shuffling the media content items in the large playlist, some conventional systems obtain identifiers (e.g., numbers such as 1 to 10,000) of the media content items and shuffle the list of numbers. Then, the conventional systems may query the database using the identifiers in the randomized order to obtain each media content item to build the randomized playlist. This too is not optimal, as it may incur significant performance costs. For example, databases are optimized for returning entries stored close together as defined by some index which sorts the entries. If there are media content items with identifiers that are distant from each other in the database, the index may not be helpful and performance of shuffling the playlist may be compromised.

We present a mechanism that solves these issues by generating a randomized playlist of a large number of media content items in an efficient manner while also presenting the media content items that are to play subsequently to the currently playing media content item in the randomized playlist. Using the mechanism, the content sharing platform can present, in the user interface on a client device, the previous and subsequent N media content items. In some

instances, when none of the previous or subsequent media content items are close with respect to the index in the database, using the mechanism may only necessitate N*2 requests to the database. As such, performance may be improved when shuffling arbitrarily large playlists of media content items using the mechanism. Also, in an implementation, the random order of the media content items in the randomized playlist may be recreated given the randomizing seed, thereby enabling stability of the shuffling, which may enhance the user experience.

The mechanism may be implemented with one or more configurable properties that balance the performance of the shuffling with the perceived randomness of the resulting randomized playlist. In an implementation, the mechanism may, rather than shuffling over the entire set of media content items in the large playlist, shuffle over smaller windows (subsets of media content items) in the large playlist. Further, as described further below, the mechanism may break the windows into smaller subwindows and randomize the order of the subwindows to enhance the randomness of the media content items in the randomized playlist. Once shuffled, a randomized playlist may be presented in a user interface on a client device where the randomized playlist may present the media content items in the random order including the previously played media content items, the currently playing media content item, and the media content items selected to play subsequent to the currently playing media content item. Such an implementation may enable a user to visualize the subsequent media content items and skip over undesired ones until a media content item in the randomized playlist is found that interests the user. Thus, the user experience may be improved from conventional systems that jump around randomly in a playlist without any indication of what is to be played next during shuffling.

Figure 1 depicts a flow diagram of a method for using the mechanism that implements a heuristic to shuffle arbitrarily large playlists in a more performant manner and to present a

resulting randomized playlist to a user, in accordance with some implementations. First, at step 102, a request for a playlist of media content items is received by a server of the content sharing platform. In the present discussion, the number of media content items in the playlist is 1,050. The user may use a user interface of the content sharing platform presented on a client device to request the playlist. The playlist may include media content items that the user has previously "liked" or media content items added manually by the user to a particular playlist, and so forth. In some implementations, the requested playlists may include a large number (e.g., hundreds, thousands, tens of thousands, etc.) of media content items (e.g., audio files, video files, etc.)

Next, at step 104, the media content items associated with the playlist may be obtained from a database. The server of the content sharing platform may query the database for some or all of the media content item associated with the playlist. Once obtained, the one or more of the media content items in the playlist may be sent to the client device for presentation to the user via a website or a stand-alone application associated with the content sharing platform.

Next, at step 106, a request to shuffle the media content items in the playlist may be received. For example, the user interface on the client device may present a "Shuffle" button and the user may select the button via an input peripheral (e.g., mouse, keyboard, touchscreen, etc.). The website or stand-alone application may receive the user input and send a request to a server of the content sharing platform to shuffle the media content items of the playlist.

Next, at step 108, the media content items are broken up into a set of windows that each includes an equal amount of media content items. "Windows" may refer to a subset of the media content items in the playlist. The number of media content items in the windows may be configurable. In some implementations, the number of media content items in the window may be the number of media content items that can be obtained from the database to satisfy desired

performance metrics. For purposes of explanation, the number of media content items in each window is set to 150 for a playlist including 1,050 media content items. It should be noted that there is a tradeoff between performance and perceived randomness to the user that is balanced via the size (e.g., number of media content items) of the windows.

For example, the media content items in a window having a smaller size may be queried faster but cause the order of the media content items to appear less random to the user because the media content items may have been originally closer to each other in the initial playlist. This concept may be explained by considering it as a distribution problem comparing window sizes of 10 and 100. If the first media content item selected has an identifier of 1, then for the media content items in the window having size 10, the probability that the next media content item has an identifier of 2 through 10 is approximately 11% (e.g., one out of the nine remaining media content items) for each media content item. In contrast, if the first media content item selected has an identifier of 1, then for the media content items in the window having size 100, the probability that the next media content item has an identifier of 2 through 100 is approximately 1% (e.g., one out of the ninety-nine remaining media content items) for each media content item. Accordingly, the probability is higher for each media content item in the smaller window size, thereby making their selection appear less random to the user.

In the present example, the first window of 150 media content items may include the following identifiers of media content items: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115,

116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150. The last window of 150 media content items may include the following identifiers of media content items: 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050.

Next, at step 110, each window may be randomized with a seed to obtain a set of windows with shuffled media content items. A seed may refer to a number that is used to initialize a randomization technique. In some implementations, the seed may be different for each window to make the randomization for each window different. For example, a seed for a first window may be 1, the seed for a second window may be 2, and so forth. The initial seed may be any suitable number and the other seeds may be lower or higher than the initial seed. The seeds for each of the different windows may proceed in a predictable manner to enable stable shuffling of the media content items in the windows. Stable shuffling may refer to the ability to recreate the random order of the media content items given the randomizing seed. In another implementation, the seed may be the same for each window thereby causing the randomization to be the same for each window. Each time the user requests to shuffle the playlist, a new seed

may be generated for the windows.

In the present example, the randomized 150 media content items in the first window may include the following identifiers: 136, 14, 56, 62, 86, 85, 133, 67, 123, 112, 55, 117, 10, 24, 8, 125, 80, 45, 2, 101, 139, 32, 137, 47, 106, 76, 102, 148, 92, 23, 52, 59, 81, 39, 132, 48, 119, 42, 122, 49, 126, 72, 110, 109, 53, 143, 82, 150, 36, 131, 116, 51, 103, 54, 145, 138, 77, 46, 100, 7, 83, 6, 147, 115, 15, 98, 61, 113, 19, 40, 18, 84, 34, 93, 134, 88, 130, 69, 75, 94, 22, 26, 3, 127, 66, 73, 142, 50, 57, 90, 35, 16, 68, 17, 58, 63, 144, 27, 21, 38, 37, 111, 29, 11, 146, 9, 140, 25, 141, 78, 65, 120, 70, 95, 118, 128, 114, 64, 87, 97, 135, 104, 124, 12, 43, 121, 20, 44, 91, 105, 1, 107, 79, 30, 74, 89, 28, 149, 71, 31, 5, 60, 13, 129, 99, 108, 33, 41, 4, 96. The randomized 150 media content items in the last window may include the following identifiers: 959, 995, 999, 936, 919, 976, 929, 985, 925, 908, 1018, 953, 911, 952, 1036, 949, 905, 1025, 1041, 957, 1050, 1022, 1013, 964, 915, 1030, 970, 968, 939, 974, 961, 1045, 1028, 982, 1017, 1001, 1004, 1032, 904, 955, 992, 984, 1031, 973, 1049, 946, 981, 1008, 1033, 960, 962, 1047, 1046, 912, 927, 1002, 926, 997, 988, 1042, 937, 951, 906, 1039, 998, 996, 902, 907, 903, 1021, 994, 987, 948, 1009, 1034, 975, 1011, 1019, 958, 1015, 967, 923, 1048, 914, 979, 972, 963, 950, 918, 1020, 1043, 940, 1027, 965, 1003, 901, 986, 942, 1000, 978, 944, 933, 956, 935, 1040, 1007, 913, 954, 1044, 1012, 941, 932, 989, 993, 947, 1035, 977, 1029, 1023, 969, 1014, 980, 934, 1005, 1026, 943, 938, 990, 917, 922, 966, 909, 910, 930, 921, 1016, 971, 931, 991, 916, 1024, 1038, 924, 1010, 928, 1037, 945, 1006, 920, 983.

Next, at block 112, each shuffled window may be broken up into a set of subwindows that each includes an equal amount of media content items. The number of media content items to include in each subwindow may be configurable to any desirable amount. In the current example, each subwindow includes 3 media content items.

The identifiers of media content items in each subwindow for the first window (1-150) may include: 136, 14, 56; 62, 86, 85; 133, 67, 123; 112, 55, 117; 10, 24, 8; 125, 80, 45; 2, 101, 139; 32, 137, 47; 106, 76, 102; 148, 92, 23; 52, 59, 81; 39, 132, 48; 119, 42, 122; 49, 126, 72; 110, 109, 53; 143, 82, 150; 36, 131, 116; 51, 103, 54; 145, 138, 77; 46, 100, 7; 83, 6, 147; 115, 15, 98; 61, 113, 19; 40, 18, 84; 34, 93, 134; 88, 130, 69; 75, 94, 22; 26, 3, 127; 66, 73, 142; 50, 57, 90; 35, 16, 68; 17, 58, 63; 144, 27, 21; 38, 37, 111; 29, 11, 146; 9, 140, 25; 141, 78, 65; 120, 70, 95; 118, 128, 114; 64, 87, 97; 135, 104, 124; 12, 43, 121; 20, 44, 91; 105, 1, 107; 79, 30, 74; 89, 28, 149; 71, 31, 5; 60, 13, 129; 99, 108, 33; 41, 4, 96. The identifiers of media content items in each subwindow for the last window (901-1050) may include: 959, 995, 999; 936, 919, 976; 929, 985, 925; 908, 1018, 953; 911, 952, 1036; 949, 905, 1025; 1041, 957, 1050; 1022, 1013, 964; 915, 1030, 970; 968, 939, 974; 961, 1045, 1028; 982, 1017, 1001; 1004, 1032, 904; 955, 992, 984; 1031, 973, 1049; 946, 981, 1008; 1033, 960, 962; 1047, 1046, 912; 927, 1002, 926; 997, 988, 1042; 937, 951, 906; 1039, 998, 996; 902, 907, 903; 1021, 994, 987; 948, 1009, 1034; 975, 1011, 1019; 958, 1015, 967; 923, 1048, 914; 979, 972, 963; 950, 918, 1020; 1043, 940, 1027; 965, 1003, 901; 986, 942, 1000; 978, 944, 933; 956, 935, 1040; 1007, 913, 954; 1044, 1012, 941; 932, 989, 993; 947, 1035, 977; 1029, 1023, 969; 1014, 980, 934; 1005, 1026, 943; 938, 990, 917; 922, 966, 909; 910, 930, 921; 1016, 971, 931; 991, 916, 1024; 1038, 924, 1010; 928, 1037, 945; 1006, 920, 983.

Next, at step114, the order of the subwindows may be randomized to obtain a randomized playlist of the media content items. Randomizing the subwindows may increase the perception of randomness of the media content items. In the current example, after shuffling the order of the subwindows, the beginning of the playlist may include the following media content items in the shuffled subwindows: 329, 374, 450; 39, 132, 48; 330, 331, 442; 822, 899, 834; 742, 733, 691,

etc.

The resulting randomized playlist may be sent to the client device and played in the shuffled order. In some implementations, a portion of the media content items in the resulting randomized playlist may be sent to the client device, thereby limiting the number of database queries that are made. At step 116, the server may cause the randomized playlist to be presented in a user interface of the website or stand-alone application associated with the content sharing platform on the client device. As noted above, the randomized playlist may present the currently playing media content item as well as subsequent media content items.

ABSTRACT

A mechanism is discussed that uses a heuristic for shuffling arbitrarily large playlists of media content items in a performant manner. The mechanism breaks up the larger playlist into smaller windows of media content items and randomizes those windows. The mechanism also breaks up the windows into smaller subwindows of media content items and shuffles the order of the subwindows. The mechanism allows customizing the tradeoff of performance and randomness to meet the behavior desired by tuning/configuring the window size and subwindow size of media content items. In addition, the mechanism enables presenting the resulting randomized playlist in a user interface such that a user may visualize the currently played media content item and any media content items that are to play subsequent to the currently played media content item.

Keywords: shuffle, random, playlist, subset, window, seed, content sharing platform, video, audio, media content item, heuristic

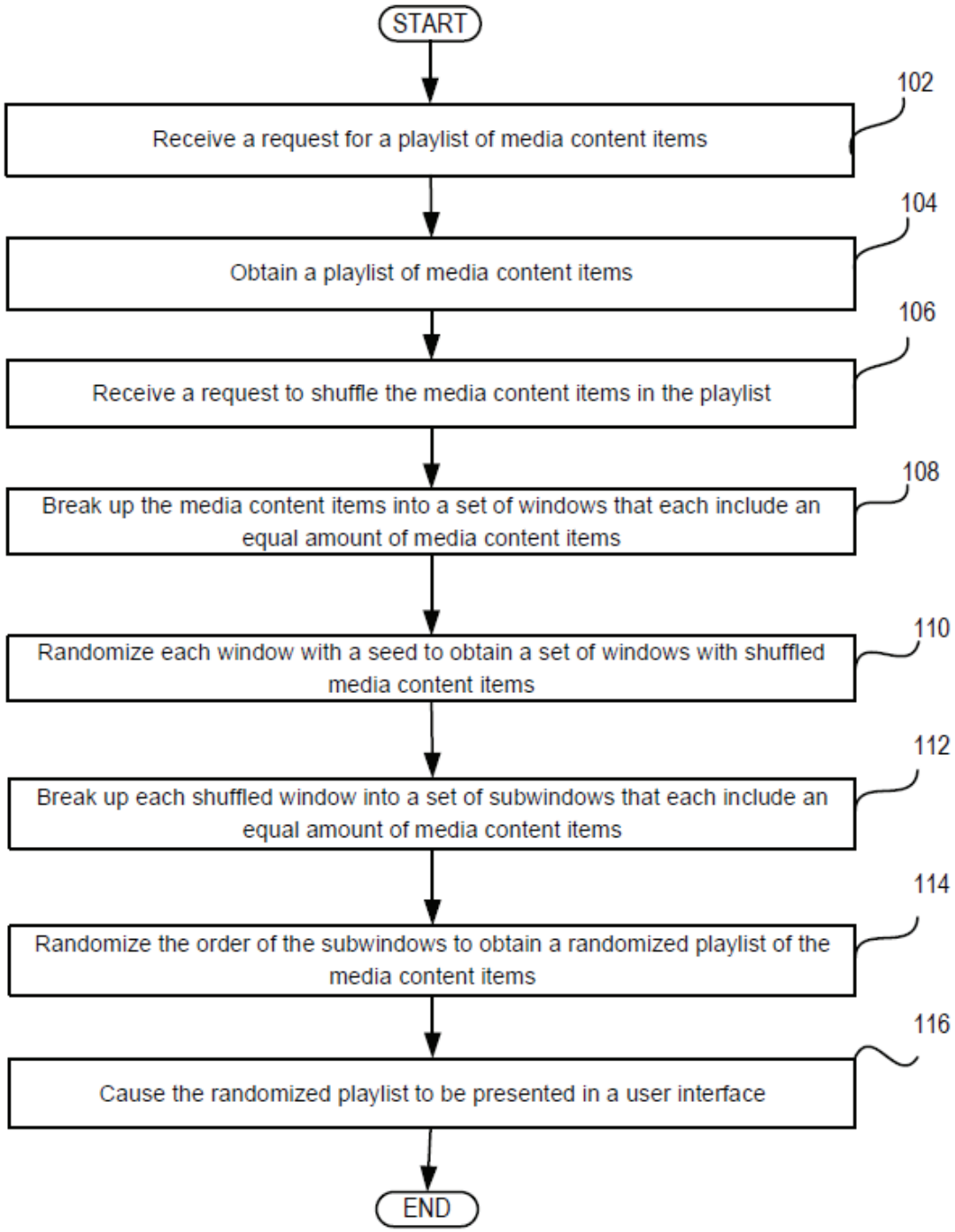
START
102
Receive a request for a playlist of media content items
104
Obtain a playlist of media content items
106
Receive a request to shuffle the media content items in the playlist
108
Break up the media content items into a set of windows that each include an equal amount of media content items
110
Randomize each window with a seed to obtain a set of windows with shuffled media content items
112
Break up each shuffled window into a set of subwindows that each include an equal amount of media content items
114
Randomize the order of the subwindows to obtain a randomized playlist of the media content items
116
Cause the randomized playlist to be presented in a user interface
END

FIG. 1