Fast Search of Audio Fingerprint using K40 GPGPU

Nguyen Mau Toan (1410215)

School of Information Science,
Japan Advanced Institute of Science and Technology

August, 2016

Keywords: Audio FingerPrint, Massively Parallel, General-purpose computing on graphics processing units (GPGPU), Hierarchy Searching, Computer System, K-modes, Locality Sensitive Hashing.

1 Introduction

Audio fingerprint is the digital fingerprint that can help to identify the audio content. The most usage of the audio fingerprint is to detect the illegal contents to support the artists to protect their copyrights. Nowadays, there are millions of audio and video contents uploaded to the internet, so the searching speed with the big database size is a big problem for all systems. Audio data can be called a form of big data and we need the special algorithm and special computer system for tackling these data. There are two main problems in this field: Extracting the audio fingerprint and searching the fingerprint in the database. The audio fingerprint extraction should identify the content even if the audio have transformed in many ways, fingerprint should be easily distinguishable from different audio waveforms (PCM) and the size of fingerprint should not be too big to store. In term of fingerprint searching, the accuracy should be considered, this depends much on not only the fingerprint extraction but also the algorithm of searching. Another important factor for audio fingerprint searching is speed, when the database is too large (about 30 million legal songs) and there are many queries (Thousands of illegal songs/tracks are updated to the Internet everyday). It is really necessary to build a computer system that can store big database and support fast search fingerprint in parallel.

2 Proposed Method: Hierarchy Massively Parallel Searching

In this thesis, we propose a new hierarchy searching system that can detect the meta information for fingerprint in real time by combining the advantages of K-modes and Locality Sensitive Hashing (LSH). Our method includes two main levels. For the first level, we cluster the database into sub-databases and these can work independently from each other for similar fingerprints belonging to the same cluster. And in the second level, we deploy searching multiple queries for local sub-database in single GPGPU device.

Our system is divided into 2 stages: Preprocessing stage and Searching stage. Preprocessing stage concentrates on building a structured database that optimizes fast searching and ability of parallelism. In Figure 1, this stage includes continuous 2 levels of clustering database and generating hash table using K-modes and LSH hash functions, respectively. The number K in K-modes will follow the number of GPGPU devices in the system. The size of each cluster after clustering must be below upper bound of the size of its GPGU's memory. To do that, we propose a new method called Extended K-modes for helping to define the limited number

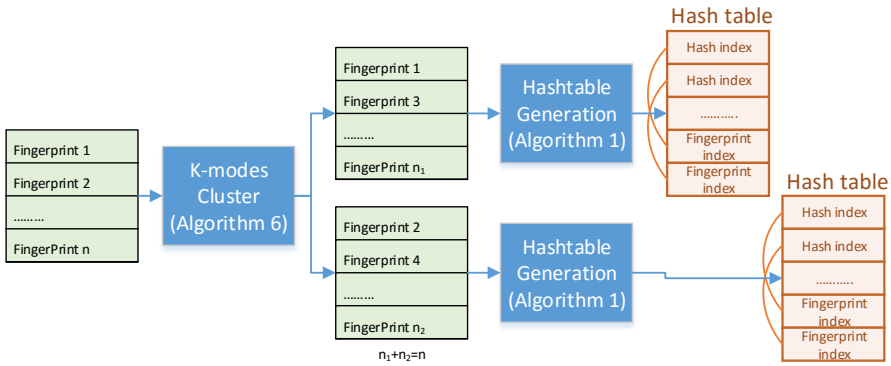

Figure 1: Preprocessing database for Hierarchy Searching for 2 devices

of the fingerprint in every cluster. With n fingerprints at the beginning, we divide into two clusters with have n_1 and n_2 fingerprints. After dividing to clusters, each cluster will be used for restructuring data by deploying buckets with the same local sensitive audio fingerprints on the same bucket.

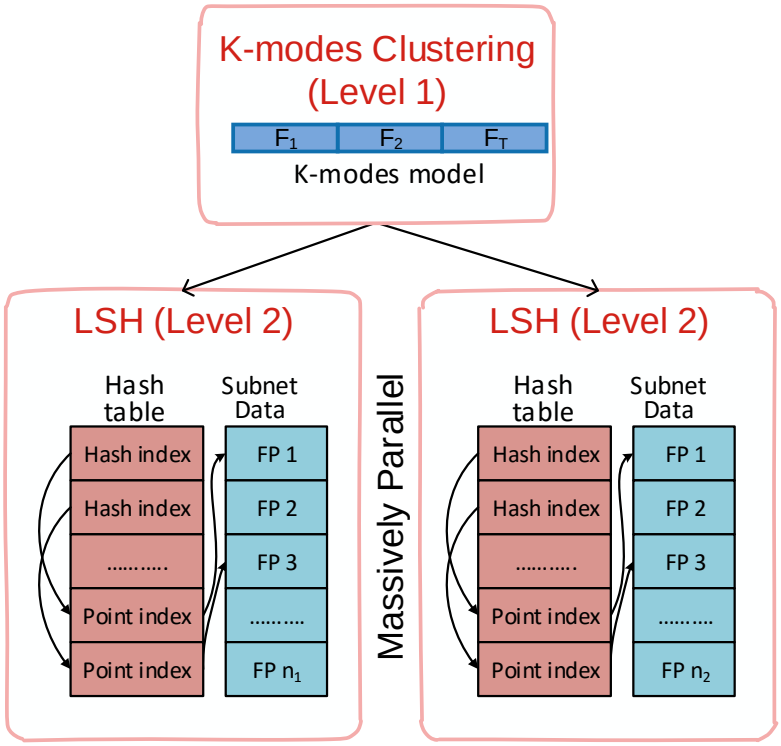

Figure 2: Overview Hierarchy Searching for Querying stage

For searching the audio fingerprint in single GPGPU we divided the queries into threads. Each query fingerprint searching has two steps, the first one is using the LSH hash functions to detect the buckets that should have its nearest fingerprint. And the second step is finding the nearest fingerprint in the specific buckets already gotten from first step. The fingerprint must be divided into 126 sub-fingerprints. The sub-fingerprint has its hash string and is pointed to a bucket. The LSH will stop when the first sub-fingerprint get the satisfied fingerprint in its bucket.

With the well-organized database in Preprocessing stage, Searching stage takes advantage of this and makes a high performance system for searching. In Figure 2, the unique Level 1 is the management of the whole system. When queries come to the system, this Level 1 will find the closest cluster for every audio fingerprint and store it to devices' queue. With the indicated audio fingerprint queries for GPGPU devices, each Level 2 (GPGPU) can self-manage its queries and its sub-database. When its queue is full, this device can copy its queries to device's global memory and start the kernel of parallel searching. Depending on the difference of a number of cores on each device, we define a queue size for better performance. After finishing one kernel, the output of fingerprint ID will be sent to Level 1. And Level 1 can send the meta information for corresponding audio fingerprint queries.

3 Evaluation

We complete building the experiment with massively parallel with 2 GPGPU devices in 1 node and test 2,000 audio fingerprint queries at the same time in disparate GPGPGU devices. Comparing with previous research of Yang in parallel audio fingerprint using FPGA, our method can achieve speed of 50 times faster when using the same conditions of database and queries. Besides, we also experiment with the audio with high distortion to meet all actual case and calculate the measure for the worst case.

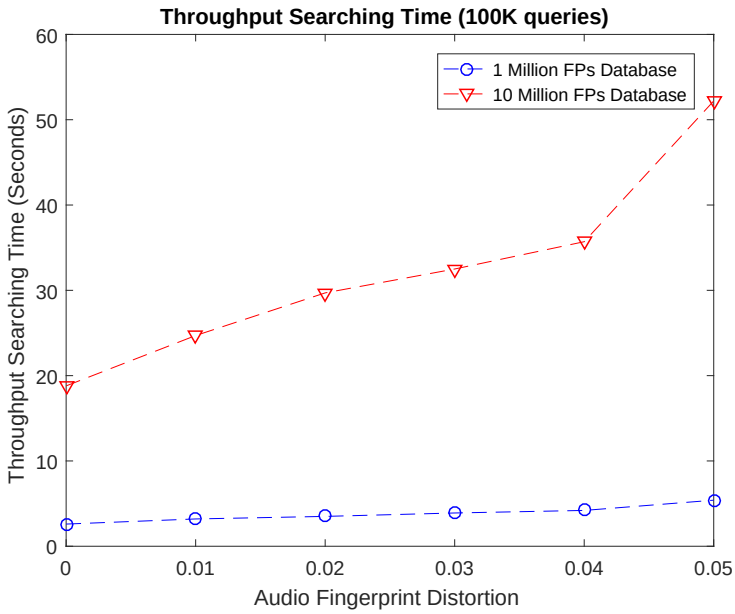

Figure 3: Result: Searching time of hierarchy searching follow the changing of database size and distortion ratio

In Figure 3, with the power of multiple GPGPU devices, we can obtain the meta information for a query within 2 milliseconds for 10 million songs' database and we can search 1000 audio fingerprint queries in parallel.

4 Conclusion and Future work

Our proposed massively parallel searching system can adapt to all GPGPUs computers. And not only does our method take advantage of all cores of GPGPUs, but it also uses CPU as the management for cluster queries and distribution queries for GPGPUs. With the searching

speed in Evaluation section, our method can be compatible with the real data of millions of songs/tracks and the searching time can meet difficult requirements of the real world's cases.

Despite many advantages listed above, our method has still several drawbacks when only focusing the searching time. First, our database is a good organization and hard to modify. It makes this database become more static database when we need to add more audio fingerprint or remove some of them. In the future, we want to re-organize the structure for storing of the hash table to become a dynamic database that support well for changing data.

The second problem is the increase of miss ratio when we increase the number of GPGPU devices. We know that the probability of miss ratio depends much on the number of devices. For this problem, we want to propose an internetwork among GPGPU devices, which will help to define a near cluster and it can be used when the miss case happens for a query in a GPGPU.