Parallel Algorithm Design for Audio Feature Extraction

Xiaocheng Luo[1, a], Jingfei Jiang[1, b], Junjie Zhu[1, c], Yong Dou[1, d]

[1]National Laboratory for Parallel and Distributed Processing, National University of Defense Technology, 109 Deya Road, Changsha, Hunan, China, 410073

[a]email: luoxiaocheng15@nudt.edu.cn, [b]email: jingfeijiang@126.com

[c]email: junjiepiglet@163.com, [d]email: yongdou@nudt.edu.cn

Keywords: Feature Extraction; Sound Recognition; Parallel Algorithm

Abstract. Audio feature extraction is a very important technique in the field of sound processing. It extremely impacts the effectiveness and correctness of sound recognition, sound verification, etc. It is a computation intensive stage in the whole sound recognition process, which is a challenging for acceleration. In this paper, a coarse-grained parallel feature extraction algorithm for high throughput of audio slices is proposed to improve the efficiency of audio feature extraction. Three typical audio feature extraction algorithms, Mel Frequency Cepstrum Coefficients(MFCC), Spectrogram image features(SIF), Octave-Based Spectral Contrast, are chosen to parallelize. Experiments results on different platforms show that the speedup of accelerated audio feature extraction is up to 17.23 on the platform with 16 cores 32 threads.

Introduction

Audio feature can represent the characteristics of a piece of music, a segment of speech and any other form of sound. They are usually used in tasks corresponding with sound processing such as speech recognition, speech synthesis, sound classification, etc. Audio feature extraction is a considerable time-consuming task. It will greatly improve efficiency of the above tasks if execution time of that can be reduced. However, it is a challenging task to parallelize feature extraction for tens of thousands of audio slices. Currently, some researchers have implemented audio feature extraction algorithm on Field Programmable Gate Array(FPGA) [1]. For example, Michal Staworko [2], Mohammed Bahoura [3], and P Ehkan [4] implemented MFCC algorithm on FPGA. Although all the above work can be satisfied with the requirements of real-time in some speaker recognition and verification system, it can hardly deal with the feature extraction of a large number of audio slices.

In this paper, we propose a coarse-grained parallel algorithm for three types of audio feature extraction, MFCC, SIF described in [5] and octave-based Spectral Contrast described in [6]. The proposed parallel algorithm focus on high throughput of audio slices feature extraction. It is a kind of task parallel algorithm and the key of its design is how to divide tasks in order to balance load and eventually obtain maximum efficiency. We analyze in detail several scheduling strategies of tasks division and find out that the default scheduling (each thread has the same tasks) gets the best performance. Experiments shows that the proposed parallel algorithm acquires desirable results.

The rest of this paper is organized as follows. The next section describes principle of three feature extraction algorithms. Then, the course of the proposed algorithm design is presented in detail. Finally, experiments are performed on different platforms to evaluate the proposed algorithm.

Audio Feature Extraction Representation

Those three algorithms are most commonly used in feature extraction. They have much in common with their execution mode and also have their respective specificity. MFCC is proposed by imitating human auditory characteristics. SIF is obtained by Fast Fourier Transform(FFT) on audio signal. In [5], SIF is treated as a real image as input of Convolutional Neural Networks(CNN) in audio event recognition. Octave-based Spectral Contrast is a discriminative feature in music type classification.

MFCC has been widely used in many speech and speaker recognition system. Compared with Linear Predictive Cepstrum Coefficients(LPCC), MFCC has better robustness and is more suitable for the human auditory characteristics. The flowchart of MFCC feature extraction is presented in Fig. 1. In the stage of pre-processing, pad is first performed on the audio signals. Then, divide the padded audio signals into overlapping frames. Fast Fourier Transform(FFT) that transforms audio signals from time domain into frequency domain is utilized to get the spectrum. MFCC will be obtained through a series of subsequent operations including Mel-scale filter, Logarithm and Discrete Cosine transform(DCT).

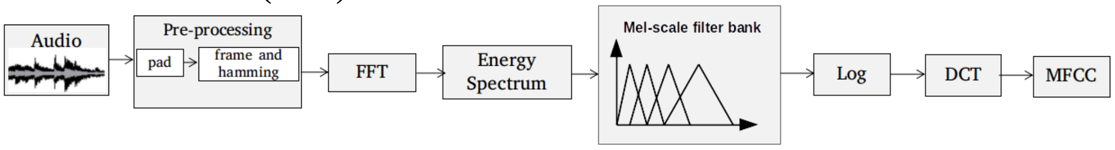


Fig. 1. The block diagram of MFCC feature extraction

SIF has been utilized successfully in some audio event recognition task. SIF imitating image processing treats frequency spectrum that is obtained by FFT as an image. The detailed SIF extraction is described in Fig. 2.

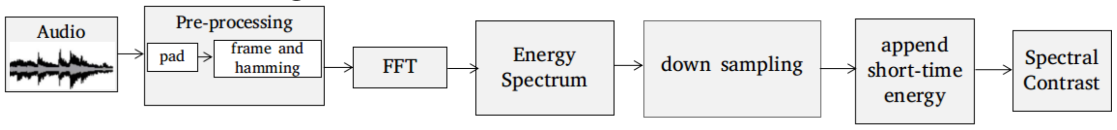


Fig. 2. The block diagram of SIF feature extraction

Given an audio signal, it is firstly split to overlapping frames on which a spectrogram is obtained by short-time Fourier transform. The short-time spectral is given by

$$S(f,t)=\left|\sum_{n=0}^{N-1} s_t(n)\phi(n)e^{\frac{-2\pi jnf}{N}}\right| \tag{1}$$

Where $f=0,\ldots,(N/2-1)$. $s_t(n)$ denotes a length-N frame at the time index t. $\phi(n)$ denotes a N-point Hamming window. the step of down sampling in the above procedure is given by

$$S_{dn}(f,t)=S(f,t)-\min_t(S(f,t)) \tag{2}$$

The short-time energy $e(t)$ can be added to the spectrogram image as an extra feature

$$e(t)=\sum_{f=0}^{F-1} S_{dn}(f,t) \tag{3}$$

Octave-based Spectral Contrast is another audio feature often utilized in music processing, such as music type classification and cover song identification. It mainly concerns the spectral peaks, spectral valleys and their differences in each sub-band. For most of the music, the strong spectral peaks approximately correspond with harmonic components while non-harmonic components, or noises often seems to be at spectral valleys. Therefore, Spectral Contrast feature could roughly represent the relative distribution of the harmonic and non-harmonic components in the spectrum. Spectral Contrast keeps abundant information and is sufficient to represent spectral characteristics of music. The detailed expressions of the spectral peaks, valleys and their differences are as follows:

$$Peak_k=\log\{\frac{1}{\alpha N}\sum_{i=1}^{\alpha N} x'_{k,i}\} \tag{4}$$

$$Valley_k=\log\{\frac{1}{\alpha N}\sum_{i=1}^{\alpha N} x'_{k,N-i+1}\} \tag{5}$$

Where the $\{x'_{k,1},x'_{k,2}\ldots,x'_{k,N}\}$ denotes sorted FFT vector in descending order, of k-th sub-band. And their difference is:

$$SC_k=Peak_k-Valley_k \tag{6}$$

Fig. 3 shows the procedure of Spectral Contrast feature extraction. The calculation of Spectral Contrast feature is roughly identical with MFCC. Spectral Contrast applies octave-scale filters in itself calculation process while mel-scale Filters are applied in MFCC. And another difference is that spectral peak, spectral valley and their difference need to be computed in MFCC, but Spectral Contrast feature does not.

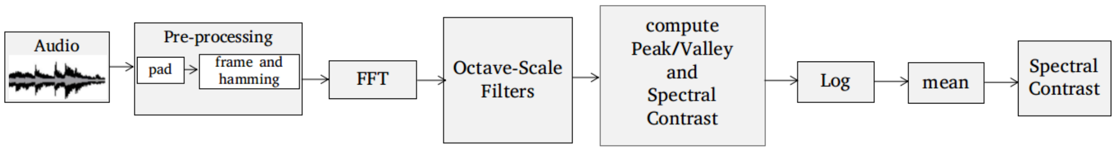


Fig. 3. The block diagram of Spectral Contrast feature extraction

Coarse-grained Parallel Algorithm

This paper focus on the feature extraction of a large amount of audio files instead of extracting feature of a single file. Therefore, the feature extraction of multifile is designed to a multithread program so that execution time will be greatly reduced. In the algorithm, each thread extracts the feature of audio files from its task queue, which is independent from each other. In the rest of this section, each part of the algorithm will be described in detail and concretely analyze scheduling strategy.

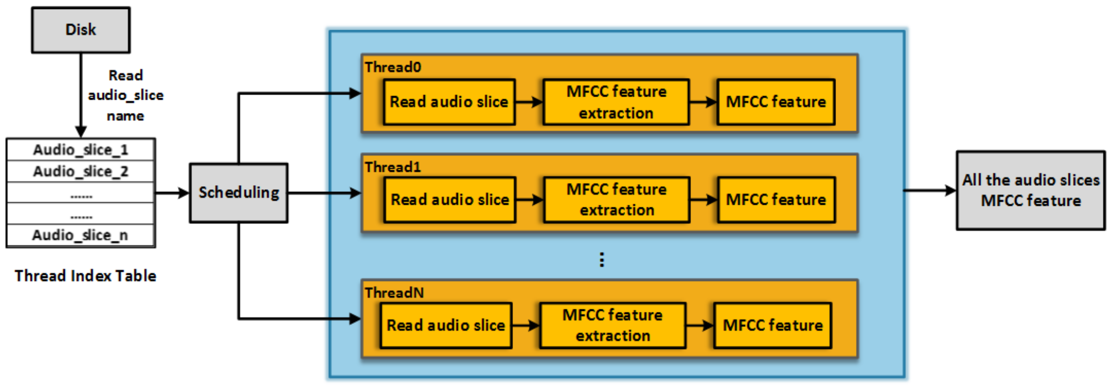


Fig.4. Coarse-grained Parallel Algorithm for Feature Extraction

The parallel algorithm for those three types feature extraction is the same. Therefore, we only show the diagram of parallel algorithm of MFCC in Fig. 4. As we can see from the diagram, the algorithm firstly read audio slices name from disk to save in a buffer named thread index table, in which thread can find where the specific audio slice is. The step of scheduling is a process of task division. Scheduling strategies need to be set manually to specify threads behavior. We should attempt to set several scheduling strategies in order to get a better performance. We mainly consider two kinds of scheduling strategies, one of which is static scheduling and the other is dynamic scheduling. Whatever strategy is static or dynamic, both of them have several different ways. The static scheduling is that the tasks of each thread have been specified before it performs. In the contrary, the tasks of each thread are determined at execution time in dynamic scheduling.

Here is an example for static scheduling. Suppose chunk_size denotes the minimum unit of assignment for task division. Suppose there are 4 threads and 100 tasks. The thread 0 will get the task number 0, 1, 8, 9… and thread 1 will get the task number 2, 3, 10, 11…, if the chunk_size=4. This task division is load balancing in this example. In the dynamic scheduling, each thread will firstly get chunk_size tasks and requires more tasks from system if and only if it finishes its tasks. However, the two kinds of scheduling have their own advantages and disadvantages. The use of static scheduling is more benefit if every task needs approximately the same computation. And using the dynamic scheduling is more benefit if the computation are great differences among tasks because the thread that has finished its own work in advance will not wait unless all the tasks have been finished. In terms of our program, we find out that there are no differences between static scheduling and dynamic scheduling. Consequently, we select the static scheduling which assign tasks evenly to each thread.

The parallel algorithm mainly exploits OpenMp to implement. During the course of algorithm implementation, third party library FFTW which is a comprehensive collection of various forms of Discrete Fourier transform(DFT) including FFT and DCT, are used to the program in order to acquire a better performance.

Experiment Results and Analysis

We select the UrbanSound8K dataset [7] for our experiments. The dataset is commonly used in sound classification. The dataset consists of 8732 audio slices of up to 4s in duration. We implement serial version as a baseline and OpenMp version to measure our parallel algorithm performance compared with previous one. The two versions of program are both tested on two experiment platforms, one of which is a personal computer(Linux, Intel Core, 4 cores 8 threads, 16GB) and the other is a server(Linux, Intel Xeon, 16 cores 32 threads, 64GB). In the implementation of program, there are 2014 data points sampled from audio slices in each per frame. The length of overlapping components between frames are set to three-quarters of one frame length. All experiments are performed on the 8732 audio slices.

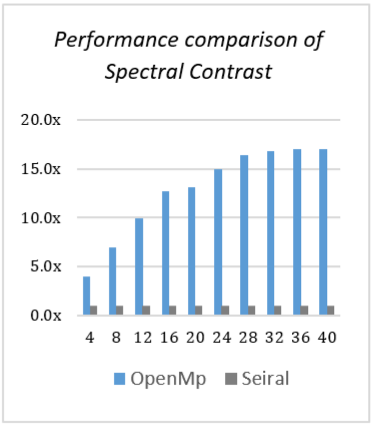
An experiment is first performed on the machine with 4 cores 8 threads. The OpenMp versions of MFCC, SIF, Spectral Contrast are executed in the case of different number of threads. The accelerated programs performance is closely related to the platform on which they are performed. They can theoretically reach the maximum speedup 4.0 or even higher running with 8 threads on the above machine. Spectral Contrast acceleration effect is reasonable, and the speedup reach to 4.54. MFCC and SIF obtain speedup 1.90 and 2.44 respectively. The detailed execution time in seconds is showed in Table 1.

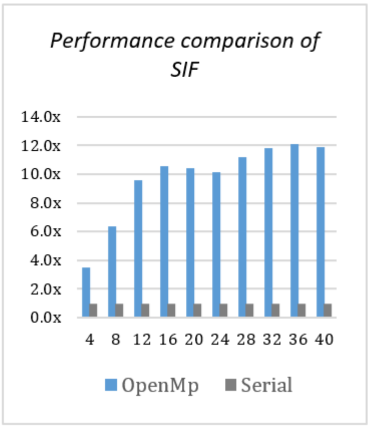
	Thread Number	2	4	6	8	10	12	14	16	18	20
MFCC	**OpenMp**	128.71	103.95	111.95	103.88	121.32	132.33	135.9	151.4	161.54	169.68
	Serial	195.81	194.19	193.64	197.77	194.35	196.45	195.91	193.86	198.03	197.35
	Sp	1.52	1.87	1.73	**1.90**	1.60	1.48	1.44	1.28	1.23	1.16
Spectral Contrast	**OpenMp**	73.18	45.04	43	31.29	34.71	39.04	38.86	37.74	38.42	36.34
	Serial	141.89	142.24	142.33	142.13	142.78	142.44	142.81	142.05	141.15	142.29
	Sp	1.94	3.16	3.31	**4.54**	4.11	3.65	3.67	3.76	3.67	3.92
SIF	**OpenMp**	36.31	26.41	26.6	25.78	26.9	27.67	27.31	27.73	27.94	27.28
	Serial	63.24	64.98	62.62	62.95	65.66	62.01	66.6	63.86	65.94	65.84
	Sp	1.74	2.46	2.35	**2.44**	2.44	2.24	2.44	2.30	2.36	2.41

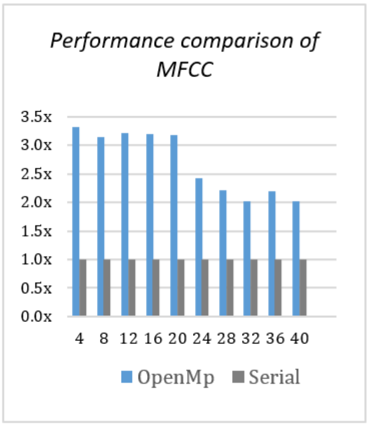
Table 1. Comparison of OpenMp and Serial versions execution time with running different number of threads

As we can see from the Table 1, Spectral Contrast obtains considerable performance while the acceleration results of MFCC and SIF features are not what we are desirable. It maybe that the use of FFTW library causes the above undesirable results. The FFTW user manual points out that there is only one thread-safe routine in it so that all other routines should be only called from one thread at a time. Besides, MFCC calls two times(FFT and DCT) FFTW in its own course of calculation and down-sampling in SIF is time-consuming. We could solve this problem in the future work.

Then, we do the same experiment on another platform, which is equipped with 16 cores 32 threads. Fig. 4 shows the results of this experiment. In the Fig. 5(a), Spectral Contrast performance is improved gradually with increasing of the number of threads and finally its speedup reaches to 17.23. It is a pretty good result. The speedup of SIF and MFCC are 12.53 and 3.36 respectively. MFCC acceleration result is not so good as SIF and Spectral Contrast due to many routines in multi-threaded FFTW are not thread safety. But the whole acceleration results are not so bad.



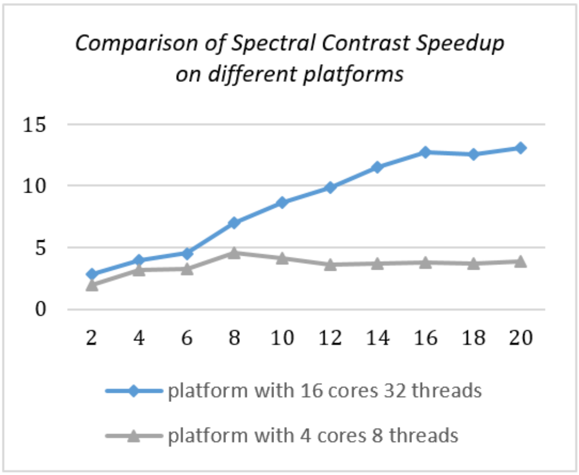




(a)The Spectral Contrast performance (b)The SIF performance (c)The MFCC performance

Fig. 5. Performance comparison of MFCC, SIF and Spectral Contrast on the platform with 16 cores 32 threads

We also make a comparison between first experiment and the second ones. Fig. 6 shows the detailed results. It is obvious that the better performance is obtained on platform with 16 cores 32 threads.



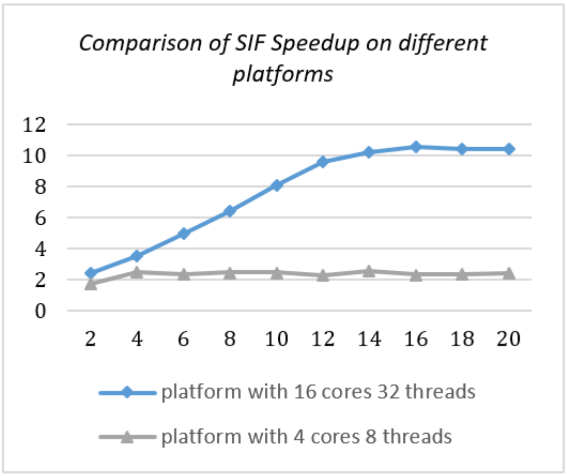


Fig. 6. Comparison of the speedup of Spectral Contrast and SIF on different platforms

Conclusion

This paper study a coarse-grained parallel algorithm for audio feature extraction. We specially parallelize the MFCC, Spectral Contrast and SIF feature for large numbers of audio slices. The proposed parallel algorithm is aim at the throughput of the feature extraction to large numbers of audio slices. In the concrete implementation of the proposed parallel algorithm, third party library FFTW is used because it is now the fastest library for computing DFT. Our programs are performed on two different platforms and the acceleration results are pretty good. The effectiveness of algorithm is showed in the experiments.

Acknowledgement

In this paper, the research was sponsored by National Science Foundation of China (Project No. 61303070 and No. 61572515) and The National Key Research and Development Program of China(Project No. 2016YFB0200401).

References

[1] Boujelben O, Bahoura M. FPGA implementation of an automatic wheezes detector based on MFCC and SVM[C] International Conference on Advanced Technologies for Signal and Image

Processing. 2016:647-650.

[2] Staworko M, Rawski M. FPGA implementation of feature extraction algorithm for speaker verification[C] Mixed Design of Integrated Circuits and Systems. IEEE Xplore, 2010:557-561.

[3] Bahoura M, Ezzaidi H. Hardware implementation of MFCC feature extraction for respiratory sounds analysis[C] International Workshop Onsystems, Signal Processing and Their Applications. 2013:226-229.

[4] Ehkan P, Zakaria F F, Warip M N M, et al. Hardware Implementation of MFCC-Based Feature Extraction for Speaker Recognition[M] Advanced Computer and Communication Engineering Technology. 2015:471-480.

[5] Phan H, Hertel L, Maass M, et al. Robust Audio Event Recognition with 1-Max Pooling Convolutional Neural Networks[J]. 2016.

[6] Jiang D N, Lu L, Zhang H J, et al. Music type classification by spectral contrast feature[C] IEEE International Conference on Multimedia and Expo, 2002. ICME '02. Proceedings. IEEE, 2002:113-116 vol.1.

[7] Salamon J, Jacoby C, Bello J P. A Dataset and Taxonomy for Urban Sound Research[C] ACM International Conference on Multimedia. ACM, 2014:1041-1044.