HIGH SENSITIVITY DETECTION OF AE EVENTS IN NOISY ENVIRONMENT USING STREAM RECORDING AND PARALLEL COMPUTATION

Kaita ITO 1, Manabu ENOKI 1

1 Department of Materials Engineering, The University of Tokyo; Tokyo, Japan
Phone: +81 3 5841 7129, Fax: +81 3 5841 7181;
e-mail: ito@rme.mm.t.u-tokyo.ac.jp, enoki@rme.mm.t.u-tokyo.ac.jp

Abstract

Conventional event-by-event recording of AE with fixed threshold voltage is often not effective in noisy environment such as monitoring of material processing. In such cases, stream recording i.e. continuous recording of AE waveforms during the whole experimental period is effective because it enables flexible and high efficient noise reduction and event detection by software. However, analysis of multi-channel AE stream often becomes a very heavy task even for modern CPU because of its extremely huge data amount. In this study, two types of processors for parallel computation i.e. CPU and GPGPU (General-Purpose computing in Graphic Processing Unit) are compared in noise reduction process in frequency domain and hit detection with multiple threshold voltages of AE streams. Optimum usage of heterogeneous multi-core processor showed high performance for analysis of AE streams. This method shows great advantage of defects monitoring during noisy environment such as dynamic material processes e.g. thermal spraying.

Keywords: Signal Processing, Continuous Waveform, Noise Filtering, Multiple Thresholds, Parallel Computing

1. Introduction

Acoustic emission method is one of the few real-time non-destructive evaluation methods for detection of many types of dynamic phenomenon in solid materials and structures e.g. micro-crack initiation and propagation, dislocation movement, friction and leakage. Conventionally in AE measurement, a threshold voltage has to be set preliminarily. An AE hit is detected when the voltage signal from AE sensor and amplifier exceeds this preset threshold voltage. This simple thresholding method needs small calculation load for AE measurement equipment. However, optimal configuration of noise filter and threshold voltage is very difficult under measurement environments with high noise level or frequent AE hits.

Therefore, AE stream recording i.e. continuous recording of AE waveform is a reliable method for successful monitoring in such environments. To this end, an AE measurement and analysis system called “Continuous Wave Memory” (CWM) has been developed [1, 2]. This CWM system includes powerful hardware e.g. computational processor and large scale storage device for continuous recording of AE streams for several days. CWM system has been also equipped powerful software for flexible real-time and post analyses of waveforms which includes frequency domain noise filter and multiple threshold voltages for event detection [3]. However, continuous recording and analyzing of multi-channel AE waveforms will induce huge data rate which is too heavy even for multi-core CPU (Central Processing Unit) for personal computers. For example, real-time processing is difficult when the data and calculation amount are huge such as noise reduction in frequency domain.

GPGPU (General Purpose computing on Graphics Processing Units) is a highly attracted solution for this problem [4]. Historically, GPU is a specialized processor for rendering of
three dimensional graphics. However, modern GPU has been evolved as SIMD (single instruction multiple data) processor with hundreds or thousands of calculation cores and it is not specialized for graphics computing. The total performance of one GPU has been reached to TFLOPS (Tera Floating-point Operations Per Second) order meanwhile CPU still remains in hundred GFLOPS (Giga-FLOPS) order. Software platform for GPGPU also has been substantiated. CUDA (Compute Unified Device Architecture) of NVIDIA® Corporation is the most widely used platform for their Tesla® and GeForce® series of processors. The OpenCL (Open Computing Language) which is an open framework for programing on heterogeneous platform also supports GPGPU. Now, GPGPU has been widely used for scientific calculations. The latest (June 2016) world-wide TOP500 list of supercomputers includes at least 66 systems with GPU acceleration [5]. Thousands units of commercial GPUs easily enables PFLOPS (Peta-FLOPS) class supercomputers with much lower cost than the originally developed many-core CPU for supercomputers. However, GPU calculation has some limitations which are not existed in the conventional CPU calculation. At first, calculation in GPU should be independent each other for better parallelization. If the calculation is sequential i.e. one calculation depends on the result of the previous calculation, these calculations cannot be parallelized. In such case, conventional CPU with small number of high performance calculation core still superior to GPU. Actually, some of TOP500 supercomputers equip only CPUs and many-core CPU accelerator units like Xeon® Phi™ from Intel® Corporation. Ultimately, adaptive heterogeneous computing with CPU and GPU is important for optimal correspondence to many types of problems.

There are some of reports which utilize GPU for processing of AE waveforms. Riha et al. used GPU to accelerate many individual functions of waveform analysis e.g. event detection, location of AE events, envelope calculation [6]. However, there are few examples about processing of AE streams and our previous study is one of the earliest reports [7]. In this study, CWM has been improved to enable real-time processing of multi-channel AE streams by using heterogeneous multi-threaded programming with CPU and GPU.

2. Experimental Procedure

2.1 Specifications of AE Measurement System

CWM system equipped Intel® Core™ i7-4770S CPU and a NVIDIA® GeForce® GT640 GPU or NVIDIA® Tesla® C2050 GPU. Specifications of these processors are shown in table 1. Hyper-threading technology was disabled in CPU for precise estimation of performance. Intel® C++ Compiler 2013, GNU C compiler version 4.6 and NVIDIA® CUDA® version 5.5 were used for building the software. In case of GPU calculation, data on the main memory which is attached to CPU have to be transferred to the GPU memory in advance. Furthermore, the data have to be transferred again to the main memory after processing in GPU.

CWM system also equipped two of Interface® PCI-3525 dual channel ADC (Analog-to-Digital Converter) boards and totally four channels of AE waveforms were continuously recorded with 10 MHz sampling frequency, 12 bits of resolution and -5 V to +5 V range. The size of one sampled data was 16 bits and the high-order 4 bits were always zero. Then, the total data rate was 80 MB/s (8 × 10⁷ bytes/s). At first, sampled data was stored in the 64 MB of first ring buffer on ADC boards, then it was sequentially transferred to the 512 MB of second ring buffer on the CPU memory by direct memory access feature of ADC boards and CPU was not used for this memory transaction. All of signal processing was done on this second ring buffer. Finally, the data was recorded to the RAID-0 storage device i.e. parallelized HDD (hard disk drives). In normal condition, two of Hitachi® 0S03361 type 7200 rpm 4 TB HDD was combined as one RAID-0 device. The read / write speed was about
250 MB/s maximum and 120 MB/s minimum. It was sufficient speed to record AE streams in real-time. However, in this study continuous AE waveform for signal processing was recorded in advance and read from the storage device. It is because the read/write speed of the storage device not to become the bottle neck on benchmarking of CPU and GPUs processing. The continuous AE waveform for testing was extracted from the waveform during plasma spraying process under very noisy condition [2]. Four of Crucial™ CT500MX200SSD1 type 500 GB SSD (Solid State Drive) was combined to be one RAID-0 device and the maximum reading speed was about 2.0 GB/s.

2.2 Procedure of Signal Processing

Figure 1 is the schematic diagram of signal processing in CWM. Performance of CPU and GPU was compared both in the noise filtering part and hit detection parts. Before and after the noise filtering, STFT (Short Time Fourier Transform) and inverse STFT processes were needed to enable a frequency domain filter i.e. arithmetic operations on time-frequency-magnitude data. FFT (Fast Fourier Transform) length in this STFT / i-STFT processing was 1,024 (2^{10}) samples. Here, STFT / i-STFT of continuous AE waveform could be parallelized i.e. one result of FFT / i-FFT does not affect to the next FFT / i-FFT. Then, the continuous AE waveform simply could be separated and assigned to each calculation thread. In this study, the length of one section was 262,144 (2^{18}) samples. Furthermore, 256 FFT / i-FFT calculation in one section was automatically parallelized in GPU by a function in CUDA. Therefore, more number of calculation threads than the number of cores in CPU and GPU could be prepared in STFT / i-STFT processing.

Meanwhile, in the hit detection part, thresholding of continuous AE waveform is difficult to be parallelized. When an AE count i.e. threshold crossing was detected, it depended on the previous result whether this count could be detected as start of an AE hit or not. Therefore, the number of calculation threads in the hit detection part was limited to (AE channels) × (the number of threshold voltages). The number of thresholds is theoretically unlimited in CWM, however, 2 dB-pitch is empirically sufficient to detect AE hits regardless of noise and signal levels of continuous AE waveform [3]. Therefore, 32 thresholds are enough to cover the whole dynamic range of AE waveform. Consequently, the number of calculation threads in the hit detection part was limited less than 128 in this study. It was much more than the number of cores in CPU but much less than GPUs.

<table>
<thead>
<tr>
<th>Table 1. Specifications of CPU and GPUs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CPU</strong></td>
</tr>
<tr>
<td>Core™ i7-4770S</td>
</tr>
<tr>
<td>Cores</td>
</tr>
<tr>
<td>Frequency / GHz</td>
</tr>
<tr>
<td>Operations / clock</td>
</tr>
<tr>
<td>Theoretical performance / GFLOPS</td>
</tr>
<tr>
<td>Memory transfer / GB/s</td>
</tr>
</tbody>
</table>

Figure 1. Schematic diagram of signal processing in CWM.
3. Results

Figure 2 shows the total throughput of CPU and GPUs in the noise filtering part with different number of calculation threads. As previously mentioned, much more parallelization was automatically enabled in GPU calculation. Calculation amount of the filtering itself was much smaller and ignorable than STFT and i-STFT. Both in case of CPU and GPU calculation, the total throughputs were improved as increasing of the number of calculation threads. However, they were eventually saturated in all cases. The maximum throughput of CPU, GPU #1 and GPU #2 were about 70, 260 and 980 MSPS (Mega samples per second), respectively.

![Figure 2. Relationship between the throughput and the number of threads in the noise filter part of the processing of continuous AE waveforms.](image)

Figure 3 shows the total throughput of CPU and GPUs in the hit detection part with different number of thresholds. CPU was much faster than GPU when the number of thresholds was small. However, the throughput of CPU was in inverse proportion to the number of thresholds whereas the throughputs of GPUs were not so diminished. GPUs showed higher throughputs than CPU when the number of thresholds was 64 or more.

![Figure 3. Relationship between the throughput and the number of thresholds in the AE hit detection part of the processing of continuous AE waveforms.](image)
4. Discussion

The throughput of noise filtering process of 4 ch × 10 MHz sampling streams could be done in real-time both by CPU and GPU. In our previous study, a previous generation CPU (Intel® Core i7-3770K, maximum 3.90 GHz, 4 cores) showed only about 40 MSPS of throughput i.e. only 60 % of current study. This significant performance improvement may be an effect of AVX2 (Advanced Vector Extension 2) new instructions of current CPU and its support by Intel® C++ compiler 2013. If AVX2 was disabled or used older generation of FFT library on compiling, the throughput was about 40 – 50 MSPS with current CPU. Meanwhile, GPU showed a significant improvement of throughputs than CPU. It is because that AE streams could be separated in many parts and processed by hundreds of calculation cores in GPU in parallel. It can be said that, STFT extracted the merit of large scale parallel computing by GPU. The bottle neck of STFT / i-STFT calculation in GPU was memory bandwidth between CPU and GPU [7]. It was only several gigabytes per second, but GPU #2 has four times of bandwidth than GPU #1 and their throughputs been proportional to their bandwidth.

The throughput of AE hit detection showed a different tendency. Thresholding of AE stream was very simple processing but could not be expected to parallelize in GPU because of its sequential processing i.e. one result affects to the next calculation. Therefore, the performance of one calculation core decided the total throughput when the number of threshold was small. In such cases, CPU had an advantage. However, when the number of threshold became large, each CPU core processed multiple thresholding simultaneously and finally GPU showed the better throughput. In this type of calculation, memory transaction between CPU and GPU was small and memory bandwidth between CPU and GPU did not be the bottle neck of calculation in GPU.

5. Conclusion

Analysis of multi-channel AE stream often becomes a very heavy task even for modern CPU because of its extremely huge data and calculation amount. In this study, two types of processors for parallel computation i.e. CPU and GPU were compared in two types of signal processing on continuously recorded AE waveforms. In general, GPU calculation showed better performance than CPU especially in case of large scale calculation for high sensitivity and automatic AE event detection in noisy environment i.e. noise filtering in frequency domain and automatic AE hit detection with many threshold voltages. Meanwhile, CPU showed sufficient performance for sequential processing. Ultimately, proper use of CPU and GPU showed the highest performance for analysis of AE streams.

Acknowledgement

We are particularly grateful for the assistance given by Mr. Dai Miyashita and Mr. Tadayoshi Okuma about development of software, and this work is supported by MEXT/JSPS KAKENHI Grant #26820302 and research subsidizing program by Moritani Scholarship Foundation.

References


