論文 / 著書情報
Article / Book Information

論題(和文)	
Title(English)	Using Graphics Hardware to Accelarate a Large Vocabulary Speech Decoder
著者(和文)	ディクソン ポール, 大西 翼, 古井 貞熙
Authors(English)	Paul Dixon, Tasuku Oonishi, Sadaoki Furui
出典(和文)	日本音響学会2008年秋季講演論文集, , No. 1-1-13, pp. 29-32
Citation(English)	, , No. 1-1-13, pp. 29-32
発行日 / Pub. date	2008, 9

Using Graphics Hardware to Accelerate a Large Vocabulary Speech Decoder*

◎ Paul R. Dixon Tasuku Oonishi Sadaoki Furui
Tokyo Institute of Technology, Tokyo, Japan
{dixonp,oonishi,furui}@furui.cs.titech.ac.jp

1 Introduction

The goal of this work was to accelerate a state-of-the-art large vocabulary speech decoder [3] by harnessing the massive computational power offered by modern graphics cards.

Modern Graphics Processor Units (GPUs) have evolved into massively parallel processors fed from specialized memory with phenomenal bandwidth. In recent years a field of research known as General Purpose GPU (GPGPU) has emerged where GPUs are used to perform general purpose computation and in many fields substantial speed improvements have been reported [10].

Recently NVIDIA released the Compute Unified Device Architecture (CUDA) [9]. This hardware and software framework massively simplifies the development of GPGPU programs. In previous generations of graphics hardware the entire pipeline, memory layout and programming tools were designed to perform only graphics operations. Developing GPGPU applications required immense skill as a knowledge of both the target application and 3D graphics programming were needed. The operations in the non-graphics algorithms would have to be replaced with graphics equivalents such as geometry operations and data would have be mapped into the graphics specific memory structures. For the CUDA generation of hardware NVIDIA introduced a platform that allows non-graphics applications to target GPUs more easily. The accompanying Application Programming Interface (API) provides a C like language which features a much less restrictive memory and input/output (I/O) model as well as support for IEEE floating point operations.

GPU performance has been increasing 2 – 2.5 times per year, whereas Central Processing Unit (CPU) performance doubles approximately every 18 months [7]. If we can move part or even all of the decoder to execute on the GPU then we can ride these faster moving performance trends. In [3] we proposed a GPGPU technique to compute the acoustic likelihoods and reported a 25% reduction in decoding time when compared to a Single Instruction Multiple Data (SIMD) CPU based approach. Even though we successfully showed that a GPU could accelerate a speech decoder one small drawback with the technique was a slowdown that occurred when using narrow beams. For narrower beams we observed the combined cost of both executing and sending data to and from the GPU was greater than performing the necessary work on the CPU. In this paper we have made massive improvement to our technique and not only achieved a significant increase in decoding speed for large models but also eliminated the slow down issues for narrower beams.

We have observed the Gaussian computations consuming a large amount of processing power, often between 50% to 70% of the total CPU time. There has been much previous research dedicated to reducing or speeding up the likelihood computations. Previous techniques include clustering, pruning and look-aheads using less complicated models [1, 5, 12, 2, 11]. However, often with approximation based techniques there will be speed accuracy trade-off. Speed gains have also been reported by utilising the vectorisation instructions available on most modern processors[4].

The remainder of the paper is structured as follows. In section 2 we first describe the GPU hardware and then describe the new GPGPU extensions that we added to our decoding engine in detail. Section 3 is a rich set of experiments designed to illustrate the speed benefits from using the GPU accelerated decoder. Section 4 concludes the paper with discussions and directions of future work.

2 Acoustic Computation using Graphics Hardware

In these evaluations we used an NVIDIA 8800GTX card that is equipped with a 128 core GPU. The GPU is implemented as a set of 16 multiprocessors, where each multiprocessor is a Single Instruction Multiple Data (SIMD) processor containing 8 cores. Each of the multiprocessor's cores executes the same instruction stream in parallel often operating on different portions of data. Thread creation on a GPU is very cheap and this is just one of the crucial factors that differentiates a GPU from a modern multi-core CPU. In fact to get maximum performance from a GPU it is necessary to launch massive amounts of threads (in the order of thousands for the current generation of hardware).

The GPU excels at certain types of computation and therefore when selecting algorithms for the GPGPU the essential first step is to select routines that will benefit most from the GPU architecture. With much more of the silicon area dedicated to computation the GPU

*グラフィックスハードウェアを用いた音声認識デコーダの高速化
ディクソン・ポール、 大西 翼、 古井 貞熙

excels at arithmetic intense computations such as dense matrix multiplication.

The acoustic model scoring is an ideal candidate for computation on the GPU because not only does the computation of acoustic model scores often account for the largest amount of CPU time during decoding [5, 13] but, the computation can be expressed as matrix multiplication [13] and this is precisely the data parallel computation task the GPU thrives on.

In our GPGPU scheme[3] all of the acoustic models are represented as a matrix $\mathbf{A}$ where each row is a representation of a log weighted Gaussian component. The i^{th} component of J dimensional mixture model is expressed as a vector of length $2J + 1$ according to:

$$\left\{ K_i, \frac{\mu_{i1}}{\sigma_{i1}^2}, \ldots, \frac{\mu_{ij}}{\sigma_{ij}^2}, -\frac{1}{2\sigma_{i1}^2}, \ldots, -\frac{1}{2\sigma_{ij}^2} \right\} \quad (1)$$

Where K_i is:

$$\log w_i - \frac{J}{2}\log 2\pi - \frac{1}{2}\sum_j \log \sigma_{ij}^2 - \frac{1}{2}\sum_j \frac{\mu_{ij}^2}{\sigma_{ij}^2} \quad (2)$$

Given an acoustic feature vector $\mathbf{x}$ the score of every Gaussian of every mixture can be calculated simultaneously by first expanding the feature vector to:

$$\mathbf{z}^T = \{1, \mathbf{x}_1, \ldots, \mathbf{x}_J, \mathbf{x}_1^2, \ldots, \mathbf{x}_J^2\} \quad (3)$$

and then performing the matrix vector multiplication $\mathbf{y} = \mathbf{Az}$. The result is a vector $\mathbf{y}$ containing log weighted scores of the feature vector for every Gaussian component for every mixture model.

The buses that connect the CPU (host) and GPU (device) together are clocked much slower and therefore a major bottleneck in the system is communicating with the GPU. In this paper we present three improvements to the GPU technique and one common theme of each technique is that they all attempt to address this communication bottleneck in some way by either reducing the size or frequency of the transfers or increasing the speed of the communication.

Batching smaller transfers into larger transfers is one of new techniques used to reduce the communication bottleneck between the device and host. In our decoder this involves sending window of acoustic frames as opposed to an individual frame to the GPU for calculation. On the device we now receive a matrix W that contains a n long windows of feature vectors in expanded form:

$$\mathbf{W} = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_n\} \quad (4)$$

The scores for every Gaussian in every mixture for the entire window are computed using the matrix multiplication $\mathbf{F} = \mathbf{AW}$, each column in $\mathbf{F}$ contains the log weighted scores for every Gaussian component of every mixture model for the corresponding feature vector in $\mathbf{W}$. Another potential advantage of the window approach is the matrix-matrix multiplication `sgemm` operation like the other level 3 Basic Linear Algebra Subprograms (BLAS) operation are often optimized more greatly than the level 1 BLAS matrix vector operations.

To reduce the size of the transfers back to the host, the GPU now also performs the *logsum* part of the acoustic computation:

$$\log p(\mathbf{x}) = \log\left(\sum_i w_i p(\mathbf{x} \mid \theta_i)\right) \quad (5)$$

Because we are dealing with the log weighted Gaussian scores $\log w_j p_j(\mathbf{x} \mid \theta_j)$ the necessary precautions are taken to avoid numerical underflows when performing the summation. The device logsum is a highly parallel implementation that makes use of the GPU hardware `log` and `exp` operations. A matrix of state mixture scores $\mathbf{M}$ is computed in parallel from the matrix Gaussian scores in $\mathbf{F}$. This gives a *window* $\times$ *mixture models* block of mixture model scores and increasing the Gaussians per mixture will not increase the size of the transfers during decoding. The host acoustic score cache logic is now redundant as we have the acoustic scores for the window in a compact block of memory. This small contiguous representation may also bring performance increases by reducing the cache misses as the acoustic scores are accessed during search, in a similar manner as described in [13, 12]. The CPU implementation used for comparison is a serial version of the logsum function that uses standard `log` and `exp` functions.

Finally by utilizing page-locked memory for the host-device transfer buffers it may be possible to obtain a further small speed-up. If a host buffer is allocated using standard page-able memory every time a transfer is initiated the CUDA drivers will allocate a temporary page-locked buffer to hold a copy of the data whilst performing the transfers

The GPGPU implementation was integrated into the decoder currently under development at Tokyo Institute of Technology[3]. The decoding engine is a time-synchronous Viterbi beam system that operates on Weighted Finite State Transducer (WFST)[8] search spaces.

To migrate the acoustic scoring portion of the decoder to execute on the GPU additional initialization and frame scoring logic is required. When the decoder is initialized the acoustic models are expanded into matrix $\mathbf{A}$ using the transform given in equation (1) and then moved onto the GPU's global memory. During the execution of the main decoding loop before searching each frame, if there are no more acoustic likelihoods in the score buffer, a window of frames is sent to the graphics card where every mixture score for the window is computed in a brute-force manner. The scores are then moved back from the graphics card into the acoustic model score buffer and used during the main token propagation search algorithm.

3 Evaluation

In this section we first describe the task, models and hardware setup. Then we present several evaluations that illustrate various aspects of the GPU decoder. In these evaluations our primary interest is the speed of the

decoder and the metric we will often use in comparisons is the Real Time Factor (RTF). When considering the speed change after enabling the GPU two metrics are used. The *speed-up factor* defined as:

$$\text{Speedup} = \frac{\text{CPU Decoding Time}}{\text{GPU Decoding Time}} \quad (6)$$

And the percent reduction in decoding time (or RTF) calculated according to:

$$\frac{\text{CPU Decoding Time} - \text{GPU Decoding Time}}{\text{CPU Decoding Time}} \times 100 \quad (7)$$

3.1 Experimental Setup

3.1.1 Corpus and Test Set

Our evaluations were carried out using the Corpus of Spontaneous Japanese (CSJ) [6]. The test set comprises of 10 lectures by male speakers only and after segmentation yielded 2328 test utterances.

3.1.2 Acoustic Features

The original 16kHz raw speech was first segmented into utterances, then converted to a 39 dimensional Mel-frequency cepstral coefficients (MFCC) based parametrizations with a 10ms frame rate and 25ms window size. Each feature vector was composed of 12 MFCC values and an energy term with delta and delta–delta values. The energy term was then discarded to give the final 38 dimensional feature representations.

3.1.3 Acoustic Models

A set of acoustic models of various complexities were built to allow us to investigate the speed characteristics of the GPU accelerated decoder. The topology of all the HMM acoustic models was a three state left-to-right tri-phone models. The state output mixture models were built using an Expectation-Maximization (EM) algorithm with mixture model splitting. There were eight sets of acoustic models each with 3000 states, the mixture models contained power-of-two sizes from two to 256 Gaussian components per mixture each with diagonal covariance.

3.1.4 Language Model

The language model was a back-off trigram using Katz smoothing with a vocabulary of 55k words trained using CSJ training data.

3.1.5 Search Network

The knowledge sources were used to create a $C \circ L \circ G$ recognition cascade containing ~3.2M states and ~7.4M arcs, the phone arcs were dynamically expanded in the decoder as explained in [3].

3.1.6 Evaluation Platform

The experiments were conducted on a 2.4GHz Intel Core2 based machine with an NVIDIA 8800GTX graphics processor. The operating system was 32bit Linux and the decoder was compiled using the GCC compiler against CUDA toolkit version 1.0.

3.2 Results

Figure (1) shows the RTF vs word accuracy curves for both the CPU and GPU accelerated decoders for beam sizes 100, 125, 150, 175 and 200 and GPU window size of 32 frames. We observe when using the same parameters both systems achieve identical accuracy indicating consistent GPU and CPU implementations. The GPU accelerated decoder is much faster than the CPU decoder. The closer bunching of the GPU curves indicates that increasing the model order is much less costly on the GPU system.

Figure (2) illustrates the speed-up factor and percent reduction in decoding time when enabling the GPU. The accelerated GPU decoder is faster than the CPU approach for all model sizes and beams widths that were evaluated, this is one of the other major advances of this technique over the results presented in [3]. The largest speed-up was for the 256 component mixtures and a beam width of 175 which translated over an 80% reduction in decoding time.

4 Conclusions and Future Work

In this paper we have described a new method for accelerating a large vocabulary speech decoder using commodity graphics processors. The approach yields a substantial improvement in decoding speed and makes using very large acoustic models possible in real-time. The other major advantages of our technique are no reduction in accuracy, exploitation of hardware that is already standard in many machines and the potential to reduce cost, space power consumption.

In the future we will continue improving our GPGPU and CPU schemes to find the optimal techniques for the respective architectures. We would also like to investigate the possibility of porting the entire decoder to run on graphics hardware as well as evaluating the suitability of GPUs for other speech pattern processing tasks.

5 Acknowledgements

This work was supported by the Japanese government METI Project “Development of Fundamental Speech Recognition Technology”.

References

[1] X.L. Aubert. Fast look-ahead pruning strategies in continuous speech recognition. In *Proc. ICASSP*, pages 659–662, 1989.

[2] A. Chan, R. Mosur, A. Rudnicky, and J. Sherwani. Four-layer categorization scheme of fast GMM computation

techniques in large vocabulary continuous speech recognition systems. In *Proc. INTERSPEECH*, pages 689–692, 2004.

[3] P. R. Dixon, D. A. Caseiro, T. Oonishi, and S. Furui. The Titech large vocabulary WFST speech recognition system. In *Proc. ASRU*, pages 1301–1304, 2007.

[4] V. Goffin, C. Allauzen, E. Bocchieri, D. Hakkani-Tur, A. Ljolje, S. Parthasarathy, M. Rahim, G. Riccardi, and M. Saraclar. The AT&T WATSON speech recognizer. In *Proc. ICASSP*, pages 1033–1036, 2005.

[5] K. Knill, M. Gales, and S. J. Young. Use of gaussian selection in large vocabulary continuous speech recognition using HMMs. In *Proc. ICSLP*, pages 470–473, 1996.

[6] K. Maekawa. Corpus of spontaneous Japanese: Its design and evaluation. In *Proc. ISCA and IEEE Workshop Spontaneous Speech Processing and Recognition*, pages 7–12, 2003.

[7] D. Manocha. General-purpose computations using graphics processors. *Computer*, 38:85–88, 2005.

[8] M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69–88, 2002.

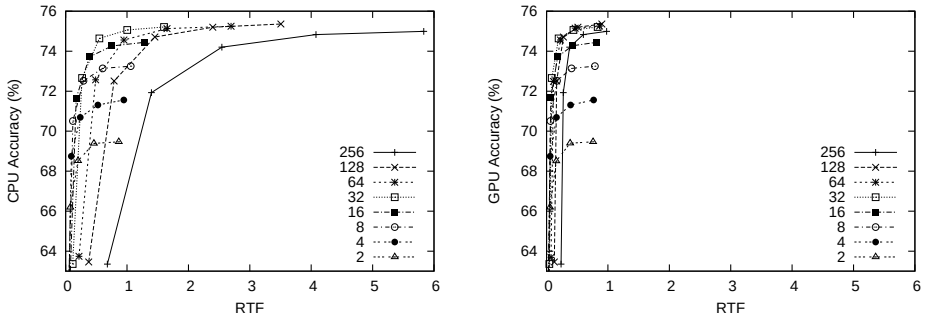
[9] NVIDIA Corporation. NVIDIA CUDA compute unified device architecture programming guide, 2007.

[10] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Kru"ger, A. E. Lefohn, and T. J. Purcell. A survey of general-purpose computation on graphics hardware. In *Eurographics 2005, State of the Art Reports*, pages 21–51, 2005.

[11] G. Saon, D. Povey, and G. Zweig. Anatomy of an extremely fast LVCSR decoder. In *Proc. INTERSPEECH*, pages 549–552, 2005.

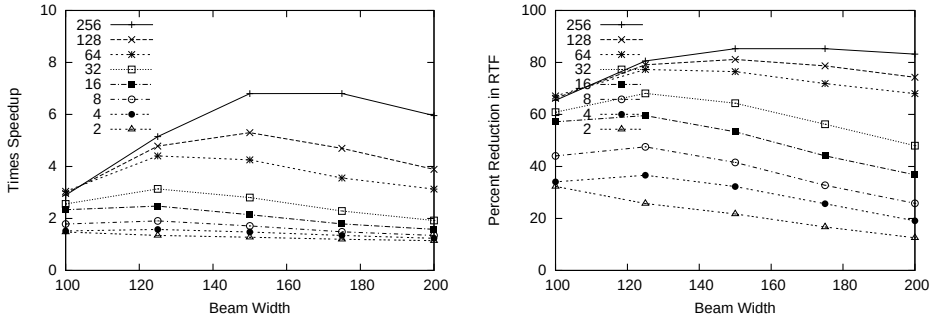
[12] G. Saon, G. Zweig, B. Kingsbury, L. Mangu, and U. Chaudhar. An architecture for rapid decoding of large vocabulary conversational speech. In *Proc. EUROSPEECH*, pages 1977–1980, 2003.

[13] M. Saraclar, M. Riley, E. Bocchieri, and V. Goffin. Towards automatic closed captioning : Low latency real time broadcast news transcription. In *Proc. ICSLP*, pages 1741–1744, 2002.



(a) CPU Implementation (b) GPU Implementation

Figure 1: Accuracy vs RTF for the CPU (a) and GPU (b) based acoustic model computations on mixture sizes from 2 to 256. Both techniques achieve the same accuracy for the same search parameters, however, the curves clearly show the GPU is substantially faster especially for large models.



(a) Times speedup from enabling GPU acceleration (b) Percent reduction in decoding time after enabling GPU acceleration

Figure 2: When enabling the GPU, the times speedup shown in (a) and percent reduction in RTF shown in Figure (b). The important result is the GPU is able to improve the decoders performance in all cases and the most significant improvements occur when using large models with large beams.