Efficient On-The-Fly Hypothesis Rescoring in a Hybrid GPU/CPU-based Large Vocabulary Continuous Speech Recognition Engine

Jungsuk Kim, Jike Chong, Ian Lane

Carnegie Mellon University

{jungsuk.kim, jike.chong}@sv.cmu.edu, lane@cs.cmu.edu

Abstract

Effectively exploiting the resources available on modern multicore and manycore processors for tasks such as large vocabulary continuous speech recognition (LVCSR) is far from trivial. While prior works have demonstrated the effectiveness of manycore graphic processing units (GPU) for high-throughput, limited vocabulary speech recognition, they are unsuitable for recognition with large acoustic and language models due to the limited 1-6GB of memory on GPUs. To overcome this limitation, we introduce a novel architecture for WFST-based LVCSR that jointly leverages manycore graphic processing units (GPU) and multicore processors (CPU) to efficiently perform recognition even when large acoustic and language models are applied. In the proposed approach, recognition is performed on the GPU using an H-level WFST, composed using a unigram language model. During decoding partial hypotheses generated over this network are rescored on-the-fly using a large language model, which resides on the CPU. By maintaining N-best hypotheses during decoding our proposed architecture obtains comparable accuracy to a standard CPU-based WFST decoder while improving decoding speed by a factor of $11\times$.

Index Terms: Large Vocabulary Continuous Speech Recognition, WFST, On-The-Fly Rescoring, Graphics Processing Units

1. Introduction

Voice user interfaces are rising as a core technology for next generation smart devices. To ensure a captivating user experience it is critical that the speech recognition engines used within these systems are robust, fast, have low latency and provides sufficient coverage over the extremely large vocabularies that the system may encounter. In order to obtain high recognition accuracy, state-of-the-art speech recognition systems for tasks such as broadcast news transcription [1, 2] or voice search [3, 4] may perform recognition with large vocabularies ($>$ 1 million words), large acoustic models (millions of model parameters), and extremely large language models (billions of n-gram entries). While these models can be applied in offline speech recognition tasks, they are impractical for real-time speech recognition due to the large computational cost required during decoding.

The use of statically compiled WFST networks, where WFSTs representing the HMM acoustic model H, context model C, pronunciation lexicon L, and language model G composed as one single network, commonly known as an H-level WFST, makes it possible to perform speech recognition very efficiently [5]. However, the composition and optimization of such search networks becomes infeasible when large models are used.

On-the-fly composition is a practical alternative to performing speech recognition with a single fully composed WFST. On-the-fly composition involves applying groups of two or more sub-WFSTs in sequence, composing them as required during decoding. One common approach is to precompose $H \circ C \circ L$ before decoding and then compose this with the grammar network G on-the-fly. On-the fly composition has been shown to be economical in terms of memory, but decoding is significantly slower than a statically compiled WFST [6].

An alternative approach for efficient WFST decoding is to perform hypothesis rescoring [3] rather then composition during search. In this approach Viterbi search is performed using $H \circ C \circ L \circ G_{uni}$, and another WFST network $G_{uni/tri}$ is used solely for rescoring hypotheses generated from the Viterbi search process in an on-the-fly fashion. Since this algorithm allows all knowledge sources are available from the beginning of the search this is effective for both selecting correct paths and pruning hypotheses.

With manycore graphic processing units (GPU) now a commodity resource, hybrid GPU/CPU computational architectures are a practical solution for many computing tasks. By leveraging the most appropriate architecture for each computational sub-task, significantly higher throughput can be achieved than by using either platform alone. Prior works [7, 8] have demonstrated the efficiency of using GPU processors for speech recognition and obtained significant improvements in throughput for limited vocabulary tasks [7]. The limited memory on these architectures, however, becomes a significant bottleneck when large acoustic and language models are applied during recognition. The most significant challenge is handling the extremely large language models used in modern broad-domain speech recognition systems [1, 2, 4]. These models can contain millions of unique vocabulary entries, billions of n-gram contexts, and can easily require 20 GB or more to store in memory. Even when significantly pruned these models cannot fit within the limited memory available on GPU platforms. To efficiently perform speech recognition with large acoustic and language models we believe a hybrid GPU/CPU architecture which leverages large memory and local-cache of the CPU with the computational throughput of GPU architectures is well suited.

In this paper we introduce a novel on-the-fly hypothesis rescoring algorithm for Hybrid GPU/CPU LVCSR engines. An overview of the proposed approach is shown in Figure 1. Viterbi search is performed on the GPU using a fully composed H-level WFST, composed with a unigram language model. For each observation frame state-likelihoods are computed locally on the GPU and when a word-boundary is encountered the partial hypotheses (word sequence) is rescored on the CPU using a large n-gram language model. State likelihoods are then updated incorporating the likelihood from the higher order langauge model, while search is progressed on the GPU.

This paper is organized as follows. In Section 2 we review

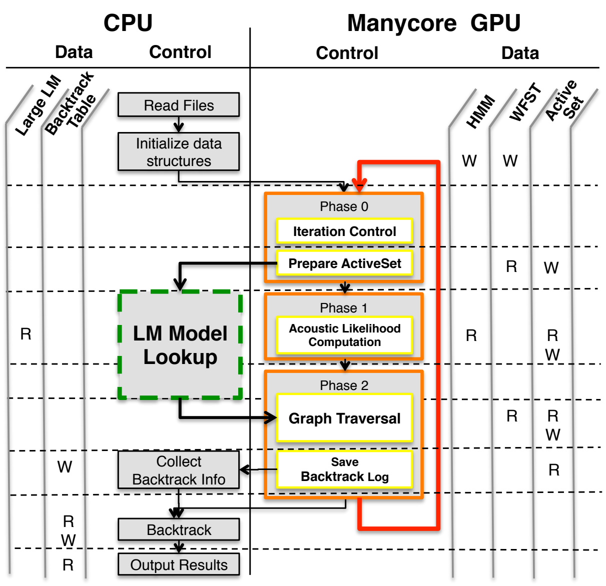

Figure 1: Detailed implementation structure.

prior works that invesitigated speech recognition decoding on manycore GPU platforms. Section 3 introduces our proposed hybrid GPU/CPU decoder and describes a novel on-the-fly hypothesis rescoring method implemented within this framework. In Section 4 we present the results of an experimental evaluation using the proposed framework, comparing the speech recognition accuracy and decoding speed obtained using CPU and GPU-based WFST decoders with our proposed GPU/CPU framework. Conclusions are given in Section 5.

2. Previous Work: GPU-based WFST Decoding

LVCSR is highly amenable for parallelization on GPUs. There are 1,000s to 10,000s concurrent tasks that can be executed at the same time in both the acoustic likelihood computation phase (Phase 1) and as well as the WFST search phase (Phase 2). In a sequential speech decoder, more than 80% of execution time is spent in Phase 1[9]. The computation in Phase 1 is a Gaussian mixture model evaluation to determine the likelihood of an input feature matching specific acoustic symbols in the speech model. The structure of the computation resembles a vector dot product evaluation, where careful structuring of the operand data structures can achieve 16-20$\times$ speedup on the GPU [7, 8]. With Phase 1 accelerated on the GPU, Phase 2 dominates the execution time. Phase 2 uses the speech model to infer the most-likely sequence seen so far.

The computation in Phase 2 involves traversal through a large graph-based model guided by acoustic symbol likelihoods. This computation has highly unpredictable control paths and data access patterns and is therefore communication intensive. Specifically, Phase 2 performs a complex graph traversal over a large irregular graph representing the WFST recognition network with millions of states and tens of millions of arcs. The data working set is too large to be cached and access pattern is determined by input available only at run time. Implementing Phase 2 on the GPU is challenging, as operations on the GPU are most efficient when performed over densely packed vectors of data. Accessing data elements from irregular data structures can cause an order of magnitude performance degradation. For

Table 1: Summary of fully composed WFST networks.

	1 Gram	2 Gram	States	Arcs
N_{bi}	4,989	835,688	3,142,982	5,748,947
N_{uni}	4,989	0	46,305	94,205

this reason, many have attempted to use the CPU for this phase of the algorithm with limited success [10, 11]. The sub-optimal performance is mainly caused by the significant overhead of communicating significant amount of intermediate results between phases 1 and 2 in every time step.

One can implement both phases of the inference engine on the GPU, and maintain the intermediate data on the GPU. This is achieved by using a technique we developed in [7, 8] for dynamically constructing efficient vector data structures at run time. This technique allows the intermediate results to be efficiently handled within the GPU subsystem and eliminates unnecessary data transfers between the CPU and the GPU. By eliminating significant data transfer overhead between the CPU and GPU, we speedup the decoding process by more than an order of magnitude compared to an optimized sequential implementation[7].

3. GPU/CPU-based on-the-fly Hypothesis Rescoring

In this paper, we propose a new on-the-fly rescoring algorithm for our Hybrid GPU/CPU based large vocabulary continuous speech recognition (LVCSR) engine. In the proposed method, Viterbi search is performed on the GPU using an H-level WFST network N_{uni}, composed using unigram language model. Higher order language models are stored and accessed via the CPU and are not composed into the H-level WFST network applied during search.

In the frame-synchronous Viterbi search, partial word hypotheses are generated by the WFST network N_{uni} when a transition outputs a word symbol. Each unique partial word hypothesis g is rescored on-the-fly using a higher order language model which resides on the CPU. Traditionally, when evaluating a transition e with an output word symbol in the WFST network, a new word hypothesis g' is generated with its likelihood, $\alpha[g']$, calculated as:

$$\alpha[g'] = \alpha[g] + \beta[e] + w[e]. \qquad (1)$$

where $\beta[e]$ is the observation probability of the input symbol, $w[e]$ is the state transition probability, and $\alpha[g]$ is the likelihood from the previous time-synchronous step.

With language model rescoring, a likelihood correction, $c[e]$, is applied:

$$\alpha[g'] = \alpha[g] + \beta[e] + w[e] + c[e, g] \qquad (2)$$

The value of $c[e, g]$ is the difference in the language model scores between the unigram language model $P_{uni}(o[e])$ used during WFST composition and a higher order language model, $P_{ngm}(o[e])$, applied during rescoring . For the word sequence, $h[g]$, that is specified in the word hypothesis:

$$c[e] = \log\left(P_{uni}(o[e])\right) - \log\left(P_{ngm}(o[e]|h[g])\right). \qquad (3)$$

The proposed rescoring process can effectively incorporate information from higher order language models. Using a WFST network based on only unigram likelihood N_{uni} is significantly

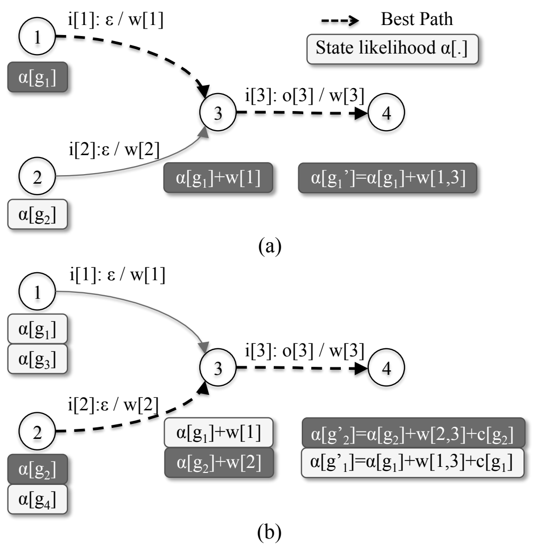

Figure 2: Example of the standard and the proposed algorithm (a) Standard algorithm (b) Proposed algorithm

smaller than a WFST network composed with a higher order language model N_{ngm}. Table 1 shows the comparative size of the WFSTs composed with unigram and bigram language model. The unigram model is approximately 1.5% the size of the bigram model even with extremely large vocabularies ($>$ 1 million words). It can easily reside in GPU memory, which is not case for higher order model.

Using a unigram WFST N_{uni} during search, however, has one major drawback. Compared to the large number of states and arcs which remain active and a WFST composed with n-gram language model (N_{ngm}), the decoding paths in the unigram case may be pruned before they reach an arc e with an output symbol where a word hypothesis can be rescored. This scenario is illustrated in Figure 2a, where the Path(2→3→4) is pruned before rescoring can be performed when evaluating the arc between states 3 and 4, even if Path(2→3→4) may be a more likely choice than Path(1→3→4).

The resulting accuracy degradation from early pruning when maintaining only the 1-best path can be observed in Figure 3. The dotted lines represents the reference accuracies when using a fully composed WFST with unigram and bigram language models. The curve with the diamond symbols shows that rescoring based on N_{uni} with a bigram language model can provide more than half of the accuracy difference, but performs significantly worse than the standard bigram case.

To resolve the early pruning issue we investigated an approach in which we maintained N-best paths when decoding the N_{uni}. As illustrated in Figure 2b, maintaining N-best paths effectively allows multiple word hypotheses to be kept until rescoring can be applied. In Figure 3, the curve with x-symbol shows that by maintaining just the 3-best paths, we obtain most of the benefit of using a bigram language model. Further more, doing rescoring by using a trigram language model for rescoring, we comfortably exceed the accuracy of an H-level WFST network composed using a bigram language model.

Maintaining N-best paths in a WFST network has many challenges. The most important challenge is the merging of N-best lists when on reconvergent paths. In Figure 2b for example, the N-best lists from State 1 and State 2 converges at State 3, and must produce an N-best list. We chose to maintain a sorted

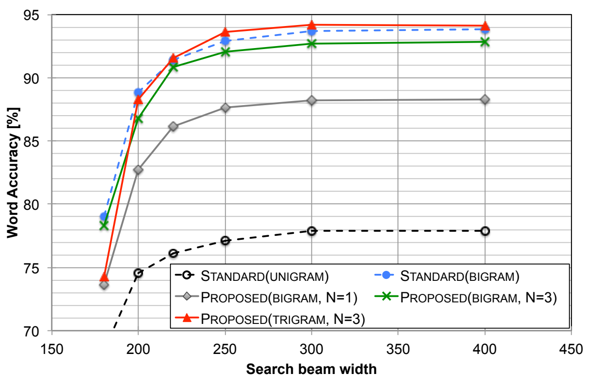

Figure 3: Relationship between word accuracy and beamwidth for maintaining different number of N-best hypothesis.

N-best list with the best path on top. This simplifies the process of merging N-best lists into a process of merge sort. On a CPU, this process can be performed quite efficiently.

On a GPU, however, there may be hundreds of arcs, each with an N-best list, trying to write into the N-best list at the destination state. We handled this challenge by developing a new technique to merge the N-best list atomically on a highly parallel platform using atomic Compare-And-Swap operations. The GPU provides hardware-supported atomic operations that are efficiently implemented and we leverage this capability to implement our N-best "merge-and-sort" algorithm on this platform.

4. Experimental Evaluation

We evaluated the effectiveness of our proposed on-the-fly hypothesis rescoring algorithm on the Wall Street Journal (WSJ) task. We evaluated the performance on the November 1992 ARPA WSJ test set which comprises of 330 sentences. The acoustic model was trained using HTK and the Wall Street Journal SI-284 data set. The resulting acoustic model contains 3,000 16-mixture Gaussians, and 39 dimensional MFCCs are used as feature vectors.

WFST networks were compiled and optimized offline. Both unigram N_{uni} and bigram N_{bi} WFST networks were optimized to the one-level step [7]. As a result, final unigram WFST network for the proposed algorithm has 98.6% smaller number of states and 98.4% smaller number of arcs compared to N_{bi} as shown in Table 1. We use the standard WSJ 5k closed language models for both WFST composition as well as the on-the-fly rescoring. For the on-the-fly rescoring, the language model probabilities are calculated on CPU using the open source toolkit KenLM [12].

We have evaluated the proposed algorithm using 2 different NVIDIA GPUs, the GTX580 (Fermi architecture) and the GTX680 (Keplar architecture) and Intel i7-2600k CPU platform. The most recent NVIDIA GTX680 GPU consists of eight new Streaming Multiprocessors (SMX) with 1536 CUDA cores. The Keplar architecture increased L2 cache hit bandwidth by 73% compared to the Fermi architecture. As a result atomic operation throughput has also been increased particularly for the atomic operation to shared address 9× times faster and the atomic operation for independent address 2.7× times faster than the GTX580 [13] .

Figure 4 shows the relationship between the word accuracy and the decoding time. We use the real time factor (RTF) as a performance measure which indicates the rate of decoding time

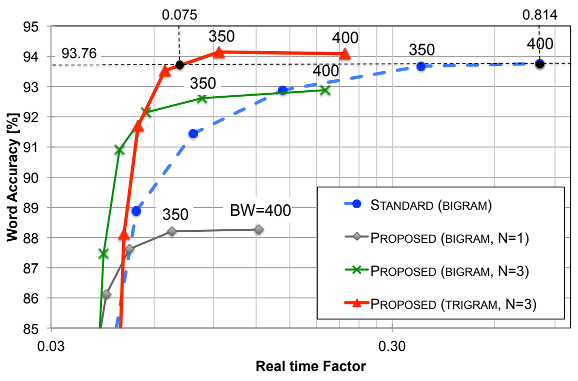

Figure 4: Relationship between word accuracy and Real Time Factor (RTF) using standard and proposed rescoring algorithms.

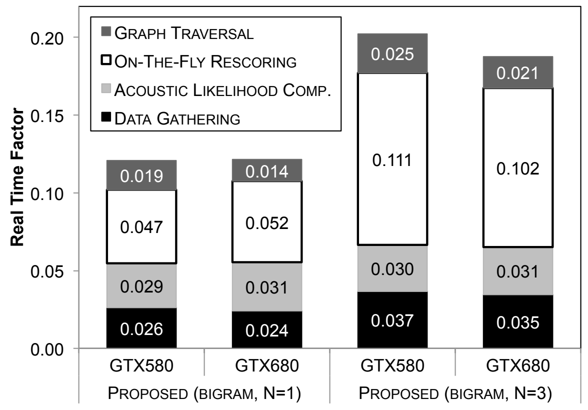

Figure 5: Ratio of the processing time per phase.

vs. utterance length. As shown in Figure 4, the proposed algorithm achieves 11× speed-up compared to standard WFST decoding method using N_{bi} at word accuracy of 93.76%. Using a trigram language model for rescoring improved word accuracy by 1.20% absolute to 94.96% while increasing the decoding time 4% compared to bigram case. We can expect further speed-up by adjusting the search beam width adaptively based on the number of active sets as proposed [7, 8].

Figure 5 shows the ratio of processing time per phase of the proposed algorithm for different size of N as well as different GPU architectures. Comparing to 1-best case, the processing time of the on-the-fly rescoring phase is increased 2× while the other phases are increased 1-1.5× when N is 3. Since language model lookup is conducted on the CPU, the processing time is increased rapidly compared to the other phases on the GPU. In case of a large N, the on-the-fly rescoring phase could be the bottleneck in our Hybrid GPU/CPU architecture. In future work we intend to investigate the use of multi-core CPUs to parallelize this phase.

Compared to the GTX580 the GTX680 performs decoding 7% faster. Major improvements are achieved from the graph traversal phase and the data gathering phase which use the atomic operation for merging and sorting the N-best hypotheses and for gathering information of the active sets. But, the acoustic likelihood computation phase achieves almost same performance with 3× more CUDA cores due to the unified clock domain an the reduced the frequency of the CUDA cores [13].

5. Conclusion

In this paper we introduced a novel architecture for a hybrid GPU/CPU-based LVCSR and proposed an on-the-fly rescoring to allow large language models to be handled at the same at as leveraging the computation performance of the modern GPU architectures. We obtain 11× speed up compared with a standard WFST decoding algorithm running on CPU at a word accuracy of 93.76%. We found that using a unigram H-level WFST network on the GPU can achieve fast execution speed and combined with on-the-fly rescoring can improve accuracy compared to the fully composed bigram case. To achieve comparable performance, however, one needs to maintain N-best hypotheses during decoding.

6. References

[1] R. Hsiao, M. Fuhs, Q. J. Y. Tam, I. Lane, and T. Schultz, *CMU/InterACT Mandarin Speech Recognition System for GALE. Handbook of Natural Language Processing and Machine Translation*. Springer, 2011, ch. 3.5.3, pp. 496–504, iSBN 978-1-4419-7712-0.

[2] U. Nallasamy, I. Lane, M. Fuhs, M. Noamany, Y. Tam, Q. Jin, and T. Schultz, *CMU/InterACT Arabic Speech Recognition System for GALE. Handbook of Natural Language Processing and Machine Translation*. Springer, 2011, ch. 3.6.4, pp. 535–540, iSBN 978-1-4419-7712-0.

[3] T. Hori, C. Hori, Y. Minami, and A. Nakamura, "Efficient WFST-Based One-Pass Decoding With On-The-Fly Hypothesis Rescoring in Extremely Large Vocabulary Continuous Speech Recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, no. 4, pp. 1352 –1365, may 2007.

[4] J. Schalkwyk, D. Beeferman, F. Beaufays, B. Byrne, C. Chelba, M. Cohen, M. Kamvar, and B. Stropek, *Google Search by Voice: A case study*. Springer, 2010.

[5] M. Mohri, F. Pereira, and M. Riley, "Weighted Finite-State Transducers in Speech Recognition," *Computer Speech and Language*, vol. 16, no. 1, pp. 69–88, 2002.

[6] T. Oonishi, P. R. Dixon, K. Iwano, and S. Furui, "Implementation and Evaluation of Fast On-The-Fly WFST Composition Algorithms," in *Proc. Interspeech*, 2008, pp. 2110–2113.

[7] J. Chong, E. Gonina, K. You, and K. Keutzer, "Exploring Recognition Network Representations for Efficient Speech Inference on Highly Parallel Platforms," in *Proc. Interspeech*, Sep. 2010, pp. 1489–1492.

[8] J. Kim, K. You, and W. Sung, "H- and C-level WFST-based Large Vocabulary Continuous Speech Recognition on Graphics Processing Units," in *IEEE Internation Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2011, pp. 1733–1736.

[9] K. You, J. Chong, Y. Yi, E. Gonina, C. J. Hughes, Y.-K. Chen, W. Sung, and K. Keutzer, "Parallel Scalability in Speech Recognition," *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 124–135, Nov. 2009.

[10] P. R. Dixon, T. Oonishi, and S. Furui, "Fast Acoustic Computations using Graphics Processors," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2009.

[11] P. Květoň and M. Novák, "Accelerating Hierarchical Acoustic Likelihood Computation on Graphics Processors," in *Proc. Interspeech*, Sep. 2010.

[12] K. Heafield, "KenLM: Faster and Smaller Language Model Queries," in *Proc. the Sixth Workshop on Statistical Machine Translation*, Edinburgh, Jul. 2011, pp. 187–197.

[13] *NVIDIA GeForce GTX680 Whitepaper*, NVIDIA, March 2012.