Exploring Recognition Network Representations for Efficient Speech Inference on Highly Parallel Platforms

Jike Chong∗†, Ekaterina Gonina∗†, Kisun You‡, Kurt Keutzer∗

∗Department of Electrical Engineering and Computer Science, University of California, Berkeley
†Parasians, LLC
‡School of Electrical Engineering, Seoul National University
jike.chong@parasians.com, egonina@eecs.berkeley.edu,
ksyou@dsp.snu.ac.kr, keutzer@eecs.berkeley.edu

Abstract

The emergence of highly parallel computing platforms is enabling new trade-offs in algorithm design for automatic speech recognition. It naturally motivates the following investigation: do the most computationally efficient sequential algorithms lead to the most computationally efficient parallel algorithms? In this paper we explore two contending recognition network representations for speech inference engines: the linear lexical model (LLM) and the weighted finite state transducer (WFST). We demonstrate that while an inference engine using the simpler LLM representation evaluates $22 \times$ more transitions per second than the advanced WFST representation, the simple structure of the LLM representation allows 4.7-6.4x faster evaluation and 53-65x faster operands gathering for each state transition. We use the 5k Wall Street Journal corpus to experiment on the NVIDIA GTX480 (Fermi) and the NVIDIA GTX285 Graphics Processing Units (GPUs), and illustrate that the performance of a speech inference engine based on the LLM representation is competitive with the WFST representation on highly parallel computing platforms.

Index Terms: Continuous Speech Recognition, Data parallel, Graphics Processing Unit, Linear Lexical Model, Weighted Finite State Transducer

1. Introduction

Highly parallel computing platforms such as the graphics processing units (GPUs) are enabling tremendous parallel computing capabilities in personal desktop and laptop systems. This shifting landscape in the underlying computing platforms needs to be taken into account during algorithm design and evaluation. In this paper we explore the use of GPUs for continuous speech recognition (CSR) on the NVIDIA GTX480 (Fermi) and the NVIDIA GTX285 GPUs. A CSR application analyzes a human utterance from a sequence of input audio waveforms to interpret and distinguish the words and sentences intended by the speaker. Its top level architecture is shown in Figure 1. The recognition process uses a language database that is can be implemented with highly efficient data parallel routines, but contains severe duplication of information in its language modeling approach. WFST is a concise representation, but its structure is highly irregular, making it difficult to execute efficiently on a data-parallel platform.

This paper explores the speed and accuracy implications of two different approaches implementing Phase 2 of the inference engine on the GPU. The first approach is based on the linear lexical model (LLM) and the second is based on the the weighted finite state transducer (WFST). LLM has a regular structure that can be implemented with highly efficient data parallel routines, whereas WFST is a concise representation, but its structure is highly irregular, making it difficult to execute efficiently on a data-parallel platform.

We demonstrate that a LLM-based inference engine can require more than an order of magnitude more computation ($22 \times$ more arc evaluations as illustrated in Section 4.2) to achieve the same recognition accuracy as a WFST-based inference engine. Its simpler structure, however, allows it to be more efficient on highly parallel GPUs. For some target accuracy, the simpler
LLM-based inference engine can execute faster than the more advanced WFST-based inference engine.

The inference engine considered in this paper uses a token-passing approach for automatic speech recognition. This type of inference engine traverses a large recognition network by keeping a set of active tokens representing alternative interpretations for the speech input. Significant research effort has focused on optimizing the representation of the recognition network used during inference. The baseline representation is the LLM. The Tree-Lexical Model (TLM) and WFST are the two widely adopted optimizations of the LLM recognition network. Tree-organization of the pronunciation lexicon reduces the number of states being traversed during recognition [5] by sharing pronunciation prefixes in a prefix tree structure. In TLM, language model lookahead can be applied for efficient pruning [6]. More recently, the WFST representation has seen wide adoption [7] because of its application of powerful finite state machine transformations to remove redundant states and arcs during offline network compilation. As reported in [8], the WFST representation is faster than the tree-lexical representation as it explores even less search space. Both TLM and WFST show faster recognition speed than conventional linear lexical search network on a sequential processor. To our knowledge, no prior research has compared the recognition speed of these network representations on highly parallel computing platforms.

Emerging multicore and manycore platforms enable speech recognition to leverage large amount of concurrency for faster recognition. There have been many attempts to parallelize speech recognition with various recognition networks on parallel platforms. Parallel implementations with the LLM-based network representations have been presented in [9, 10]. Parallel implementations with the TLM-based networks on multicore platforms are described in [11, 12]. WFST-based speech recognizers are also parallelized in [13, 14]. Recently, GPUs have been adopted for parallelizing speech recognition applications. Much of the effort is focused on computing acoustic likelihoods in parallel because this phase is computationally intensive and embeds significant fine-grained concurrency [3, 4]. Recently, some progress has been made on parallelizing the communication intensive phase (i.e. the Viterbi search). A complete data parallel LVCSR on the GPU with a LLM-based recognition network was presented in [2]. Parallel WFST-based LVCSR is also implemented on CPU and GPU in [14, 15]. [14] compared sequential and parallel implementations of the WFST-based recognition network representations. This paper contrasts the implications of using LLM and WFST recognition network representations on the GPU. In the following sections, we will briefly introduce the key differences in the recognition network representations as well as outline our implementation strategies to arrive at efficient parallel implementations.

2. Network Structure

The network structures for the LLM and the WFST representations differ significantly. As illustrated in Figure 2, the LLM representation is constructed by providing a chain of triphone states for each pronunciation to be recognized. For example, a dictionary of 5,000 word pronunciations would have 5,000 chains of triphone states in the recognition network. There are many duplications as each pronunciation chain is constructed using a separate copy of the triphone states for the chain’s phone sequence. The possibility of sharing common prefix is not considered in the LLM representation. The language model captures the likelihood of word-to-word transitions. It is used when the token-passing recognition process reaches the end of a chain of triphones. Since a word can be followed by any word, one must evaluate the possibility of transitions from one word to all words in the vocabulary.

In contrast, the WFST representation of the recognition network is constructed by composing the pronunciation model and the language model using powerful finite state machine (FSM) composition techniques. Both within-word transitions and across-word transitions are explicitly represented in the composed network. The sparsity structure of natural languages and the minimization techniques employed in [7] allow the WFST representation to avoid the state explosion problem when composing the models. As reported in [7, 8], the WFST representation is a concise representation that encapsulates a large amount of information with little redundancy. As we will show in our results, compared to the LLM representation, many fewer tokens are required to be maintained for the WFST representation during inference to achieve a target recognition accuracy. The separate pronunciation and language models in the LLM representation allow for highly optimized computation kernel design, compensating for the performance lost from the redundancies in the representation.

3. Traversal Implementation

The two inference engines are implemented on the GPU, with one using the LLM representation of the recognition network and the other using the WFST representation. Both inference engines are designed using the structure shown in Figure 3, where both the observation probability computation and the token passing phases are implemented on the GPU. This structure has been shown to provide the best recognition speed on GPUs for both types of recognition networks [2, 14]. The implementations are designed to utilize hardware support for atomic operations on the GPU. In Phase 1, each Gaussian Mixture Model (GMM) in the acoustic model is evaluated in a thread-block; in Phase 2, each arc in the recognition network is evaluated by a thread. Backtracking is performed on the CPU. We highlight the key differences between the graph traversal process on recognition networks using the LLM representation and the WFST representation.

The LLM representation: Efficient implementation of the graph traversal of a network with LLM representation depends on explicitly handling two types of transitions in the LLM representation: within-word transitions and across-word transitions. This distinction was originally used in a parallel implementation by [9], and is elaborated here to include three specialized data structures for the GPU implementation. The three struc-
The acoustic model was trained by HTK [16] with the speaker independent training data in the Wall Street Journal 1 corpus. The frontend uses 39 dimensional features that have 13 dimensional MFCC, delta and acceleration coefficients. The trained acoustic model consists of 3,000 16-mixture Gaussians. Two language model sizes are used to illustrate the effect of a reduced language model size on the recognition performance. The language model weight is set to 15. The WFST network is an $H \circ C \circ L \circ G$ model compiled and optimized offline with the dmake tool described in [17]. The test set consists of 330 sentences totaling 2,404 seconds from the Wall Street Journal test and evaluation set. The serial reference implementation using the LLM representation has a word error rate (WER) of 8.0% and runs with a 0.64 real time factor. The number of states and arcs representing the recognition networks used are shown in Table 1. WFST representation explicitly stores all across-word transitions, and LLM representation skips many across-word transitions as they are implied by the format. This leads to the differences in the actual number of arcs stored.

Table 1: Recognition network sizes used in experiments

<table>
<thead>
<tr>
<th>LLM Plumed</th>
<th>LLM</th>
<th>WFST Plumed</th>
<th>WFST</th>
</tr>
</thead>
<tbody>
<tr>
<td># State</td>
<td>122,246</td>
<td>122,246</td>
<td>1,081,285</td>
</tr>
<tr>
<td># Arcs</td>
<td>537,608</td>
<td>1,596,884</td>
<td>2395,145</td>
</tr>
</tbody>
</table>

4.2. Computation Load, Accuracy, and Speed

We measured the number of transitions evaluated by an inference engine using both the LLM and the WFST representations of the recognition network. As illustrated in Figure 4(a), each curve represents a set of recognition accuracy results by adjusting the average number of states maintained active. A dynamic threshold prediction scheme is used for pruning. We chose an operating point at 8.90% WER for the following comparison with an 11% relaxation (8.0% to 8.9%) in accuracy to gain almost a doubling in decoding speed (from 7.4x to 13.7x faster than real time for the WFST representation on GTX480).

At 8.90% WER, by using the LLM representation, an inference engine must evaluate 22x more transitions compared to using the WFST representation. For a sequential implementation where the execution is usually compute-bound, using the WFST representation could provide a significant speed advantage over the LLM representation.

The speed and accuracy trade-off shifts significantly for implementations on highly parallel platforms. Figure 4(b) illustrates the speed of the speech inference engine on the GTX285. The speed is shown as *Real Time Factor*, which is the number of seconds required to process one-second of speech input. The simplicity of the LLM representation allows the inference process to surpass the speed of the WFST representation for target WER of more than 8.8%.

Figure 4(c) illustrates the speed of the speech inference engine on the GTX480. Going from GTX285 to GTX480, the new processor microarchitecture provided an average of 25% speed improvement for the LLM representation, and 79% improvement for the WFST representation. On the GTX-480, with the availability of data caches, the WFST representation becomes
While this analysis is based on a 5K vocabulary recognition network, larger models are expected to benefit more significantly from the highly parallel computing platforms. Work is in progress to examine the recognition network representation trade-offs for larger models. Both GTX285 and GTX480 maintain an abstraction of uniform memory access latency. As other manycore processor architectures emerge with non-uniform memory access architecture (NUMA), one may want to experiment with static or dynamic partitioning of the recognition network to increase locality [9, 14]. With careful management of data working set, factoring of the WFST network could also be explored to increase locality [7].

6. References


Figure 4: WER w.r.t. # of transitions evaluated (a), execution time in Real Time Factor (b/c), and speed analysis at 8.90% WER (d)