GPU-Accelerated Large Vocabulary Continuous Speech Recognition for Scalable Distributed Speech Recognition

Jungsuk Kim  Ian Lane
Electrical and Computer Engineering
Carnegie Mellon University

March 20, 2015 @GTC2015
Overview

• Introduction
• Background
  • Weighted Finite State Transducers in Speech Recognition
• Proposed Approach
  • GPU-Accelerated scalable DSR
• Evaluation
• Conclusion
Introduction

• Voice interfaces a core technology for User Interaction
  • Mobile devices, Smart TVs, In-Vehicle Systems, …

• For a captivating User Experience, Voice UI must be:
  • **Robust**
    • Acoustic robustness ➔ Large Acoustic Models
    • Linguistics robustness ➔ Large Vocabulary Recognition
  • **Responsive**
    • Low latency ➔ Faster than real-time search
  • **Adaptive**
    • User and Task adaptation
Introduction

• Large models critical for accurate speech recognition:
  • Large acoustic models \(\Rightarrow\) Tens of Millions of parameters
  • Large vocabulary \(\Rightarrow\) Millions of words
  • Large language model \(\Rightarrow\) Billions of n-gram entries (\(\geq 20\text{GB}\))

• Examples include:
  • Acoustic modeling for telephony [Mass 2014] or Youtube [Bacchiani 2014]
    • \(\sim 200\text{M}\) parameter Deep Neural Networks
  • Language model rescoring for Voice Search [Schalkwyk 2010]
    • 1.2M vocabulary, 5-gram LM, \(12.7\text{B}\) n-gram entries
Introduction

Speech recognition contains many highly parallel tasks

Large Models
More Accurate

Graphic Processing Units
(SIMT, ~3000 cores, <24GB)
optimized for parallel computing

= ASR engine designed specifically for GPUs
Introduction

- 1 Million Vocabulary (3-gram)
- 30 Million parameter Deep Neural Network

<table>
<thead>
<tr>
<th>GPU</th>
<th>RTF (s)</th>
<th>xRT</th>
<th>1hour (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tesla K40</td>
<td>0.02</td>
<td>50X</td>
<td>72s</td>
</tr>
<tr>
<td>Titan X</td>
<td>0.01</td>
<td>100X</td>
<td>36s</td>
</tr>
<tr>
<td>Tegra K1</td>
<td>0.17</td>
<td>6X</td>
<td>612s</td>
</tr>
<tr>
<td>Tegra X1</td>
<td>0.14</td>
<td>7X</td>
<td>504s</td>
</tr>
</tbody>
</table>

Specifications:
- Tesla K40: Kepler, 2880 cores
- Titan X: Maxwell, 3072 cores
- Tegra K1: Kepler, 192 cores
- Tegra X1: Maxwell, 256 cores
Background

*Weighted Finite State Transducers (WFSTs)* in *Speech Recognition*
WFST in Speech Recognition

- "Recognize speech" v.s. "Wreck a nice beach"...
- Search is performed in 3 phases.
  - Phase 0: Active Set Preparation.
  - Phase 1: Acoustic Score Computation.
  - Phase 2: WFST Search.
• **Phase 0: Active Set Preparation**
  • Collect active hypotheses from previous frame.
WFST in Speech Recognition

• **Phase 1: Acoustic Score Computation**
  • Compute acoustic similarity between given speech and phonetic models using Deep Neural Network
**WFST in Speech Recognition**

- **Phase 2: WFST Search**
  - Perform frame synchronous Viterbi beam search on WFST network.
  - If multiple transitions have same next state \( s \), then the most likely (minimum score) hypothesis is retained (i.e. state 12, 14, 15...)
WFST in Speech Recognition

- Iterate these 3 phases until input audio ends.
- **Phase 0:** Active Set Preparation
WFST in Speech Recognition

- **Phase 1: Acoustic Score Computation**
WFST in Speech Recognition

- **Phase 2: WFST Search**
• **Phase 0: Active Set Preparation**
WFST in Speech Recognition

- **Phase 1: Acoustic Score Computation**
WFST in Speech Recognition

- **Phase 2: WFST Search**
Recognized result is an output symbol sequence over the best path.

Result: “RECOGNIZE SPEECH”
Proposed Approach

GPU-Accelerated Scalable DSR
Distributed Speech Recognition (DSR)

- **Iteration control**
  - Allocate or deallocate data structures.
  - Terminate decoding task.

- **Feature extraction**
  - Receive audio and extract feature for current iteration (batch).
  - Speaker dependent adaptation.

- **Acoustic score computation**
  - Deep Neural Network (Forward Propagation).

- **Graph search**
  - Conduct frame synchronous WFST search.
  - End-of-utterance detection.

- **Post processing**
  - Output (Lattice) processing.
  - Sending result back to client.
Producer/Consumer design pattern

- Master/Slave pattern.
- Decuple processes that produce and consume data at different rates.

**Advantages:**
- Enhanced data sharing
- Processes can run in different speeds.
- Buffered communication between processes.
Architecture 1 (Naïve)

(0) Iteration control, data preparation, result handling.

(1) Extract features from active audio-streams into stacked feature vector

(2) Stack incoming frames from active audio-streams and compute likelihoods

(3) Conduct Viterbi beam search over WFST and conduct on-the-fly rescoring

(4) Send result back over TCP/IP, Data-collection.

Producer thread

Audio-Stream 1 ($A_1$)

Audio-Stream N ($A_N$)

Shared data structures and models

Feature Extraction → Acoustic Score Computation → Graph Search → Post Processing

Feature Extraction → Acoustic Score Computation → Graph Search → Post Processing

Feature Extraction → Acoustic Score Computation → Graph Search → Post Processing

DFFFT Transform And Filterbanks → Acoustic Model → Decoding Graph Language Model
Architecture 1 (Naïve)

• **Pros.**
  • *Maximum decoding performance.*
  • Simple thread management.

• **Cons.**
  • Low throughput and GPU utilization if batch size is small.
  • Number of consumer threads can be limited by GPU (by maximum inflight kernels)
  • Not suitable for many CPU + single GPU configuration.
Architecture 2

(0) Iteration control, data preparation.

(1) Extract features from active audio-streams into stacked feature vector

(2) Stack incoming frames from active audio-streams and compute likelihoods

(3) Conduct Viterbi beam search over WFST and conduct on-the-fly rescoring

(4) Send result back over TCP/IP, Data-collection.

Update hyp. (H₁)

Update hyp. (Hₙ)
Architecture 2

• Pros.
  • More scalable and configurable structure.
  • Can assign more threads to bottleneck phase.
  • Interleaving frames from multiple tasks.
  • Can achieve maximum utilization of GPU.

• Cons.
  • Complex threads configuration.
  • More queuing overheads
  • Expected relatively higher latency compared to “structure 1”
Evaluation Results

GPU-Accelerated Scalable DSR
Evaluation Setup

- **Language Model:**
  - 1 Million Vocab. 3-gram (10.1M n-gram)

- **Acoustic Model:**
  - DNN: (in) $253 \times 2048 \times 2048 \times 2048 \times 2048 \times 2048 \times 3432$ (out)

- **Feature type:**
  - 23$^{th}$ Filterbank coefficient with CMVN

- **Evaluation Set:**
  - WSJ eval92 (20K, 333 utts.)

- **Platform:**
  - Core i7-2600K + NVIDIA Tesla K40
**Evaluation Results**

- **GPU only configuration** (G1, G2): 1 Tesla K40.
- Architecture 2 improves speed by **0.24 RTF** (N=35)
- “**Architecture 2**” processes **35** concurrent audio streams in real-time.

**WER: 6.7%**

<table>
<thead>
<tr>
<th>Process Type</th>
<th>G1</th>
<th>G2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration control</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>ASR decoder</td>
<td>2*</td>
<td>0</td>
</tr>
<tr>
<td>Feature extraction</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Acoustic score comp.</td>
<td>0</td>
<td>1*</td>
</tr>
<tr>
<td>Graph search</td>
<td>0</td>
<td>2*</td>
</tr>
<tr>
<td>Post processing</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

* use GPU
Evaluation Results

- **Hybrid configuration** (H2): 1 GPU + 2 CPU (16 cores).
- “Architecture 2” processes **80** concurrent audio streams in real-time.
Conclusion

GPU-Accelerated Scalable DSR
Conclusions

• Proposed **scalable** and **configurable** DSR server architecture.
• “**Architecture 2**” was able to process ...
  • **40** concurrent audio streams in real-time with 1 GPU (K40c)
  • **80** concurrent audio streams in real-time with 1 GPU + 16 CPU cores.
• **Performance can be improved further**
  • Lock-free task queue.
  • Optimal / Adaptive Thread configuration.
  • Smart task scheduling.
References
References


Q&A

Thank you for your attention.