LVCSR System on a Hybrid GPU-CPU Embedded Platform for Real-Time Dialog Applications

Alexei V. Ivanov
Educational Testing Service
90 New Montgomery St.
San Francisco, CA, USA
alexei_v_ivanov@ieee.org

Patrick L. Lange
Educational Testing Service
90 New Montgomery St.
San Francisco, CA, USA
plange@ets.org

David Suendermann-Oeft
Educational Testing Service
90 New Montgomery St.
San Francisco, CA, USA
suendermann-oeft@ets.org

Abstract

We present the implementation of a large-vocabulary continuous speech recognition (LVCSR) system on NVIDIA's Tegra K1 hyprid GPU-CPU embedded platform. The system is trained on a standard 1000-hour corpus, LibriSpeech, features a trigram WFST-based language model, and achieves state-of-the-art recognition accuracy. The fact that the system is real-time-able and consumes less than 7.5 watts peak makes the system perfectly suitable for fast, but precise, offline spoken dialog applications, such as in robotics, portable gaming devices, or in-car systems.

1 Introduction

Many of nowadays' spoken dialog systems are distributed systems whose major components, such as speech recognition, spoken language understanding, and dialog managers, are located in the cloud (Suendermann, 2011). For example, interactive voice response (IVR) systems are often connected to conventional telephony networks and handle a substantial portion of customer service interactions for numerous organizations and enterprises. One of the advantages of cloud-based systems is the strong computational power such systems can have which is believed to be critical for some of the components to produce an adequate performance (see for example recent advances in commercial speech recognition systems (Hannun et al., 2014)).

Despite their advantages, cloud-based spoken dialog systems have several limitations. E.g. they not only real-time-able speech recognizers, which poses a number of additional constraints to the implementation of the system (Ivanov et al., 2016), but, first and foremost, they require a high-speed, high-reliability, and high-fidelity connection to the client device. If this precondition is not met, spoken dialog systems cease to be what they promise to be: dialog systems. Slow, clunky, and intermittent connections may be acceptable with pseudo-dialog applications such as the ones typical in virtual assistants (Suendermann-Oeft, 2013), but they are not suited for realistic conversational applications such as in customer care (Acomb et al., 2007), virtual tutoring (Litman and Silliman, 2004), or command and control. Even more importantly, there are numerous applications for spoken dialog systems which require operation in offline mode altogether, for example in moving vehicles (Pellom et al., 2001), with robots in adverse conditions (Toptsis et al., 2004), in certain medical devices (Williams et al., 2011), or with portable video game consoles and toys (Sporka et al., 2006).

Maintaining a cloud application server farm, capable of supporting the mass service comes at a recurring operational cost, which limits the range of possible revenue models with which the spoken dialog system can be offered. A way to solve this problem is to transfer the necessary hardware to the client device and let the customer naturally cover the processing power costs. Thus, reduction of the complexity of the involved technology and reducing its power consumption become critical figures of merit according to which the portable systems such as robots, portable game consoles, and toys are going to compete.

Further advantages of using a low-footprint highly accurate real-time able speech recognizer over cloud-based recognition include

- no need for complex load balancing, instance management, or distributed, redundant server architectures;
- lower energy footprint due to the elimination of server and communication hardware

Proceedings of the SIGDIAL 2016 Conference, pages 220–223,
Los Angeles, USA, 13-15 September 2016.

needed to run cloud-based speech recognition jobs;

- no privacy concern since the data remains on the local hardware (which can be important for applications in medical, intelligence, defense, legal, or financial domains, among others);

- straight-forward user adaptation directly on the client hardware without the need to maintain potentially millions of customer profiles in the cloud;

- reduced network activity resulting in lower operation costs and improved bandwidth for other concurrent tasks requiring network communication, especially for wireless applications;

- enhanced options for voice activity detection since the speech recognizer can be constantly running, while constant streaming of audio from a client to the cloud is not feasible.

In Section 2 we present a large vocabulary speech recognition system architecture designed for NVIDIA's Tegra K1 hybrid GPU-CPU embedded System-on-a-Chip (SoC). We show that its recognition accuracy performs on par with state-of-the-art systems while maintaining low power consumption and real-time ability in Section 3.

2 System Description

Research has shown that an interaction with a dialog application becomes overly tiresome for the human interlocutor when the system's response does not occur promptly (Fried and Edmondson, 2006; Wennerstrom and Siege, 2003; Shigemitsu, 2005). Speech recognition is only the first of many steps in producing the system's response. Therefore, it is crucial that the recognition output can be produced at the rate of speech or as close to that as possible. While compromising recognition accuracy for a better real-time factor (xRT) is trivial, maintaining the state-of-the-art performance within the real-time constraints is challenging (Ivanov et al., 2016).

Building on top of our results described in (Ivanov et al., 2015) we implemented a highly parallel real-time speech recognition inference engine with rapid speaker adaptation that is model-level compatible with the Kaldi toolkit (Povey et al., 2011). It maintains the state-of-the-art accuracy while doing real-time online recognition with the NVIDIA's Tegra K1 hybrid GPU-CPU embedded platform. The recent studies (Morbini et al., 2013; Gaida et al., 2014) confirm state-of-the-art level of the Kaldi model-generation pipelines.

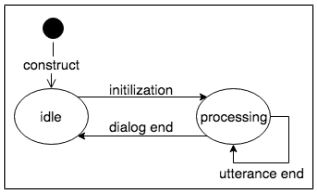

Figure 1: State diagram

Figures 1, 2 and 3 show the ASR architecture we designed. The interaction with the ASR system follows the Client-Server architecture. Figure 1 depicts the states the server transitions through from the client's perspective. After start-up of the ASR system, it stays in an idle state until a client opens a session with the server. This session is implemented as a web-socket connection. If the ASR system is used as a component within a dialog application, this session stays open until the full conversation with the human user is finished. When the server receives a new session, it transitions into the processing state, in which it immediately processes the incoming audio on a per-chunk basis. Finalizing recognition of a single utterance within the dialog is triggered by an "end of the utterance" signal. The upstream dialog system component receives the recognition result either on a per utterance basis or as a partial feedback up until the current position in the utterance. Ability to interactively produce the intermediate recognition results is an essential feature of the dialog-oriented speech recognizer as it allows us to start interpretation of the user input even before its completion. The recognition session is stopped when the client closes the web-socket connection. Then, the server transitions back into the idle state.

In the processing state, the data flows through the ASR system as shown in Figure 2. We grouped the individual steps in the employed ASR pipeline, namely: audio data acquisition, feature extraction, i-Vector computation, acoustic probability computation, decoding, backtracking and propagating the result back to the client, represented as blocks within the figure into the modules that run in a single thread. The modules are connected to each other via the ring buffers represented as wide ar-

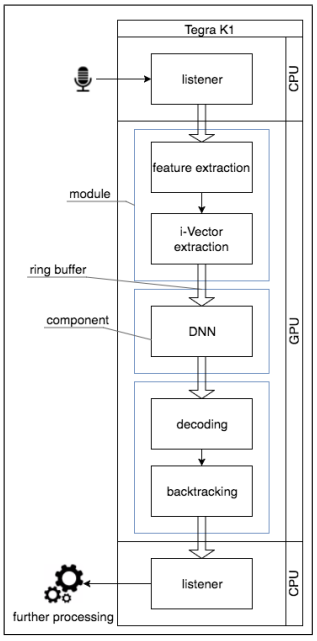

Figure 2: Data flow diagram

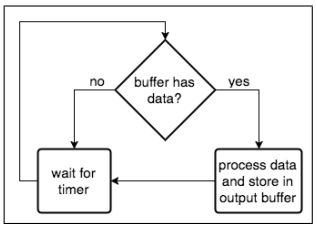

Figure 3: Component diagram

rows. Each module operates as shown in Figure 3. Every x seconds, the module checks if the input ring buffer contains a new data chunk, processes the chunk, stores it in the output ring buffer and repeats.

In Figure 2, the backtracking component is grouped together with the decoding component and executed in each processing cycle. This setup allows the ASR system to generate a partial result for each input chunk. A possible alternative to this strategy is to trigger the backtracking by the 'end of utterance' signal and only compute the resulting lattice once. This would additionally save computation time and is useful when there is no need for partial results during the recognition process.

The components running in modules placed on the GPU part of the chip have been especially implemented to utilize the parallel computing advantages of GPUs. Processing speedup with GPUs is achieved via committing larger areas of the die for solving the single task. Compared to CPUs graphical processing units (GPUs) allow for an easier processing resource management. The GPU chip lacks extensive control logic making it potentially more efficient. The downside is increase of the programming effort.

3 Experiments

In order to verify our design we have used a set of the models, generated by the standard Kaldi model-generating recipe for the LibriSpeech acoustic corpus (Panayotov et al., 2015). Specifically, we have used the Deep Neural Network – Weighted Finite State Transducer (DNN-WFST) hybrid with i-vector acoustic adaptation. The acoustic model is implemented as the 8-layer p-norm DNN with approximately 14.22 million free parameters stored as single precision floating point numbers. For language modeling we have taken a version of the standard LibriSpeech tri-gram model pruned with the threshold of $3\mathrm{e}{-7}$. There are approximately 200 thousands uni-grams, 1 million bi-grams and 34 thousands tri-grams. The resulting WFST has the complexity of about 10 millions nodes and 25 millions arcs. The i-vector is evaluated from a separately trained Universal Background Gaussian-Mixture Model (UBM-GMM) with 512 Gaussians. The final i-vector has 100 components. The standard Kaldi MFCC features are used.

The evaluation has been performed with the standard LibriSpeech test sets, namely: "DC" - 2703 clean development recordings ($\approx$ 5h. 24 min.); "DN" - 2864 noisy development recordings ($\approx$ 5h. 8 min.); "TC" - 2620 clean test recordings ($\approx$ 5h. 25 min.); "TN" - 2939 noisy test recordings ($\approx$ 5h. 21 min.). The evaluation is performed as the single-pass recognition with online acoustic adaptation within the speaker-specific utterance sets in order to simulate operation of the speech recognizer in short single-user dialogues.

We compare performance of our Tegra-based speech engine with the reference implementation of the Kaldi online2-wav-nnet2-latgen-faster decoder that is running on a system powered by the Intel Core i7-4930K CPU at 3.40GHz clock frequency. All operating parameters (the pruning beam widths, model mixing coefficients, etc.) are kept the same between the reference and the presented system. Table 1 summarizes accuracy and real-time factors of the compared systems. There was a minor random WER difference between the GPU and CPU implementations, similar to what was reported in the earlier publications (Ivanov et

Tasks	WER, %	CPU 1/xRT	TK1 1/xRT
DC	≈ 7.2	1.11	1.20
DN	≈ 19.6	0.97	1.02
TC	≈ 7.8	1.11	1.15
TN	≈ 19.4	0.95	1.01

Table 1: Accuracy and speed of compared recognizers. WER – word error rate; "CPU 1/xRT" – the inverse of the real-time factor (i.e. the processing-production speed ratio) for the reference system; 'TK1 1/xRT" – inverse real-time factor for the presented system. Power consumption is 150W for the "CPU" and 7.5W for the "TK1" systems. The Tegra system hardware cost is approximately 10 times smaller.

al., 2015). The WER figures reported in the table reflect the average expected performance level.

4 Conclusions

We have demonstrated the possibility to achieve the state-of-the-art accuracy with a dialog-oriented real-time able speech recognition inference engine running on a low-power consumer-grade SoC. The presented system implements a single-pass recognizer with online speaker-adaptation which is essential in dialogs. The system is immediately usable to support rich dialog experience with the guaranteed low latency locally-run dialog systems that take advantage of the complex large vocabulary continuous speech recognition models.

References

K. Acomb, J. Bloom, K. Dayanidhi, P. Hunter, P. Krogh, E. Levin, and R. Pieraccini. 2007. Technical Support Dialog Systems: Issues, Problems, and Solutions. In *Proc. of the HLT-NAACL*, Rochester, USA.

J. Fried and R. Edmondson. 2006. How Customer Perceived Latency Measures Success In Voice Self-Service. *Business Communications Review*, 36(3).

C. Gaida, P. L. Lange, R. Petrick, P. Proba, A. Malatawy, and D. Suendermann-Oeft. 2014. Comparing Open-Source Speech Recognition Toolkits. Technical report, DHBW Stuttgart, Stuttgart, Germany.

A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Ng. 2014. Deep speech: Scaling up end-to-end speech recognition. In *in Proc: of the ICSLP*, ArXiv, 1412(5567).

A. V. Ivanov, P. L. Lange, and D. Suendermann-Oeft. 2015. Fast and power efficient hardware-accelerated cloud-based asr for remote dialog applications. In *in Proc. of ASRU'2015*, Scottsdale, AZ, USA.

A. V. Ivanov, P. L. Lange, D. Suendermann-Oeft, V. Ramanarayanan, Y. Qian, Z. Yu, and J. Tao. 2016. Speed vs. accuracy: Designing an optimal asr system for spontaneous non-native speech in a real-time application. In *Proc. of the IWSDS*, Saariselk, Finland.

D. Litman and S. Silliman. 2004. ITSPOKE: An Intelligent Tutoring Spoken Dialogue System. In *in Proc: of the HLT-NAACL*, Boston, USA.

F. Morbini, K. Audhkhasi, K. Sagae, R. Artstein, D. Can, P. Georgiou, S. Narayanan, A. Leuski, and D. Traum. 2013. Which ASR should I choose for my dialogue system. In *Proc. of the SIGDIAL*, Metz, France.

V. Panayotov, G. Chen, D. Povey, and S. Khudanpur. 2015. LibriSpeech: an ASR corpus based on public domain audio books. In *in Proc. of the IEEE ICASSP*, Brisbane, Australia.

B. Pellom, W. Ward, J. Hansen, R. Cole, K. Hacioglu, J. Zhang, X. Yu, and S. Pradhan. 2001. University of Colorado Dialog Systems for Travel and Navigation. In *in Proc: of the HLT*, San Diego, USA.

D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely. 2011. The Kaldi Speech Recognition Toolkit. In *Proc. of the ASRU*, Hawaii, USA.

Y. Shigemitsu. 2005. Different Interpretations of Pauses in Natural Conversation - Japanese, Chinese and Americans. *Academic Report*, 27(2).

A. Sporka, S. Kurniawan, M. Mahmud, and P. Slavik. 2006. Non-speech input and speech recognition for real-time control of computer games. In *in Proc: of the Assets*, Portland, USA.

D. Suendermann-Oeft. 2013. Modern conversational agents. In J. Jähnert and C. Förster, editors, *Technologien fuer digitale Innovationen: Interdisziplinaere Beitraege zur Informationsverarbeitung*. Springer VS, Wiesbaden, Germany.

D. Suendermann. 2011. *Advances in Commercial Deployment of Spoken Dialog Systems*. Springer, New York, USA.

I. Toptsis, S. Li, B. Wrede, and G. Fink. 2004. A multimodal dialog system for a mobile robot. In *in Proc: of the ICSLP*, Jeju, South Korea.

A. Wennerstrom and A. F. Siege. 2003. Keeping the Floor in Multiparty Conversations: Intonation, Syntax, and Pause. *Discourse Processes*, 36(2).

J. Williams, S. Witt-Ehsani, A. Liska, and D. Suendermann. 2011. Speech Recognition in a Multi-Modal Health Care Application: Two Sides of the Coin. In *Proc. of the AVIxD/IxDA Workshop*, New York, USA.