Speech Recognition on Cell Broadband Engine

[Extended Abstract]

Yang Liu
Lawrence Livermore Nat'l Lab
7000 East Ave
Livermore, CA 94550
liu24@llnl.gov

Holger Jones
Lawrence Livermore Nat'l Lab
7000 East Ave
Livermore, CA 94550
jones19@llnl.gov

Michael Perrone
IBM Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598
mpp@us.ibm.com

ABSTRACT

Speech processing involves several computationally intensive tasks that may benefit from hardware acceleration on streaming computer architectures. However, identifying and exploiting parallelism opportunities in this computation represents a significant challenge, especially given the scope of the problem and the data formats necessary in order to maximize the hardware resources. We present in this poster our design, implementation, and first results of a connected-digit speech recognition system on the Cell Broadband Engine architecture using the TI-DIGITS corpus. The key components of this basic system are implemented as efficient and compact kernels for the SPE processors and are reusable for more sophisticated speech recognition systems. Our early benchmarking efforts indicate that the performance of this prototype system is 64 times faster than than the reference ISIP software implementation, which emphasizes the Cell's potential for full speech processing. This poster also identifies some next steps in this work such as applying subword (phone) models to accomodate larger vocabulary sizes, and incorporating language grammar models and rules to reduce the overall complexity of this computation.

1. INTRODUCTION

Speech processing has already been successfully incorporated into many application areas and commercial products on portable and embedded devices, despite their limited processing capabilities. However, other applications, such as telephony for automatic call centers, which receive and must process thousands of speech channels, have much larger requirements in computation and recognition accuracy. These requirements continue to grow and will soon exceed the processing capabilities of commercially available systems. The traditional approach of distributing this computation across a set of computer nodes is not a viable solution in the long term since compute clusters generally do not scale well – they are expensive to build and maintain, and other considerations such as physical space and power consumption rates will limit their practical deployment. However, recent trends have shown that streaming architectures, such as the Cell Broadband Engine, designed for task or data-level parallelism, have better cost-performance than traditional computer architectures optimized for instruction-level parallelism. At 3.2 GHz, the Cell processor has a peak performance of 230 GFLOPS, which is roughly 9 times greater than the performance of a dual-core 3.2 GHz Pentium processor. Furthermore, the Cell's communication model also allows memory-bound applications to conceal data transfer lantencies. Speech recognition often involves a great deal of computation and data management, which suggests that Cell might be an efficient architecture for performing this type of processing. In this study, we implement a prototype connected-digit speech recognition on Cell to see how much of its computational potential can be leveraged for speech processing.

2. CELL PROCESSOR

The Cell processor contains a Power Processing Engine (PPE) and eight Synergistic Processing Engines (SPEs). The PPE is a basic PowerPC core with a VMX unit. The PPE executes the operating system and is typically responsible for directing the activities of SPEs, where the the bulk of the parallel computation is performed. Each SPE consists of a 128-bit register file with 128 registers and 256K of local memory, and two in-order dual-issue pipelines. The large register file simplifies register renaming operations to allow higher levels of compiler optimization. The 128-bit register can be used to represent different vector sets of bytes, half-words, and words. Each SPE also provides SIMD arithmetic and logic instructions (SPE intrinsics) to operate over these vectors and achieve multiple operations per clock.

Data transactions between SPEs and the PPE are conducted via DMA transfers, which may be initiated from the PPE or a SPE. By utilizing predictable data access patterns, DMA transfer latencies can be effectively concealed using double-buffering techniques to pipeline memory movement with instruction execution. This is an important advantage over traditional computer architectures, which instead relies upon large caches to achieve performance. Programs for the PPE and SPE are written and compiled separately and may be linked either statically at compile-time or dynamically at runtime. SPE programs may take advantage of the vector instruction sets and dual-issue pipeline, while PPE programs typically organize and direct the computation. The Cell API uses a threaded interface to manage SPE programs,

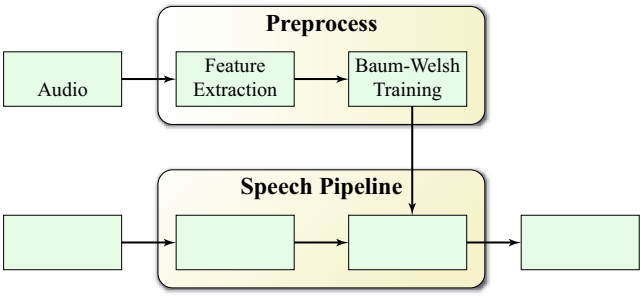

Figure 1: A real-time speech recognition system generally consists of two parts: feature extraction and decoding. A complete speech processing system also includes training, but this step is usually very computationally expensive and typically performed in an offline process.

and atomic semaphor instructions manage synchronization between SPE-SPE and SPE-PPE to provide flexibility in implementing the necessary communication models.

3. SPEECH RECOGNITION

A speech recognition system 1 extracts features from an audio channel and then applies pattern-matching to identify units of speech, which are then further parsed into words and sentences. The feature extraction front-end generates a sequence of feature vectors from an audio channel to capture unique and/or general spectral and temporal properties of the signal. The extracted features are typically based upon the Mel Frequency Cepstral Coefficients (MFCCs) – a signal representation standard adopted by the speech recognition community. The pattern-matching part of this system is implemented using a Hidden Markov Model (HMM), which is a finite state machine (FSM) that models the stochastic processes involved in speech synthesis. HMMs can directly model units of speech such as phones, tri-phones, or words, but require a significant amount of transcribed training data. The HMM decodes speech by computing a maximum-likelihood path computation through the FSM. This path is computed by the Viterbi and Level-Building algorithm, which are based upon dynamic programming.

4. CELL IMPLEMENTATION

The MFCC feature extraction front-end is implemented by a set of signal processing filters in a SPE program. Each feature vector represents a 25 ms frame of speech, with 15 ms of overlap with its neighboring frame (e.g. 100 feature vectors are generated from 1 s of speech). Audio data must be packed into quad-words in order to achieve maximum parallelism in the computation.

The Vitebi algorithm is implemented by two SPE programs. The first, *decode obs* computes only the observational probabilities from the decoded MFCCs, and the second, *decode max* computes the maximum likelihood using the computed observational probabilities. This factorization is useful because the observational probabilities need only be computed once in a single pass, while the Level Building algorithm (implemented on the PPE) may require several iterations over the maximum likelihood computation. Within

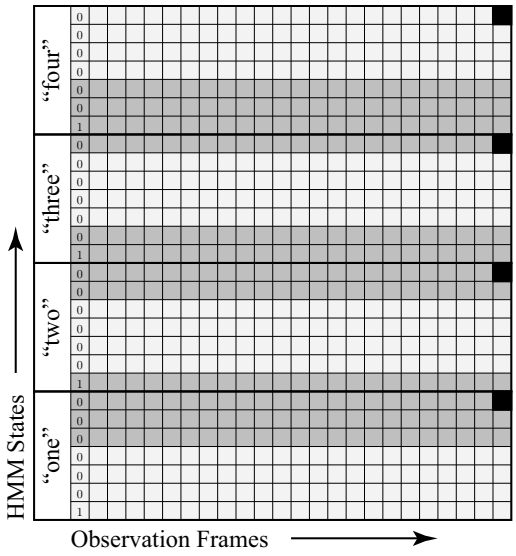

Figure 2: The Viterbi algorithm is implemented using SIMD parallelism by stacking multiple HMMs together.

the *decode_obs* and *decode_max* kernels, HMMs are stacked 2 together to achieve SIMD parallelism. HMM boundaries are delinated using the state transitional probabilities to prevent values in one HMM from contaminating its neighbor.

5. RESULTS

Our prototype system recognizes ten unique digits ("zero" through "nine"), along with "oh" and "(sil)", modeled by 12 HMMs. We use a Gaussian Mixture Model, with 4 Gaussian kernels to characterize the observational probabilities at each state (we have plans to tie Gaussian parameters together to reduce the space and time complexity of the decoding). The system decodes up to four digits at a time at a rate of approximately **1,500 channels per second** (1,500 seconds of audio time per second of processing time). Our system performs 64 times faster than the ISIP software implementation, which only decodes audio at 23 channels per second. Future work includes implementing phone models to expand the vocabulary size and hierarchical pruning strategies to reduce the total amount of computation.

APPENDIX
A. ADDITIONAL AUTHORS

John Johnson (johnson22@llnl.gov)
Sheila Vaidya (vaidya1@llnl.gov)
Borivoj Tydlitat (borivoj tydlitat@cz.ibm.com)
Ashwini Nanda (ashwini@us.ibm.com)