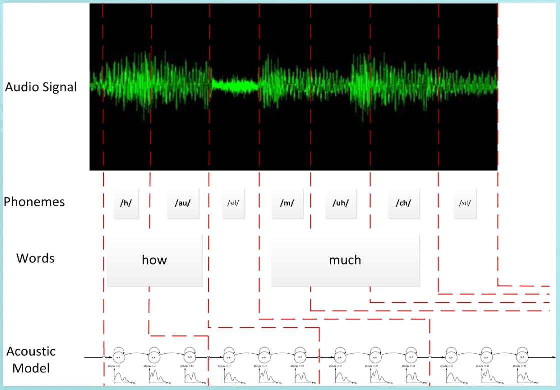
Rapid Training of Acoustic Models Using GPUs

Senaka Buthpitiya, Ian Lane, Jike Chong
Department of Electrical and Computer Engineering, Carnegie Mellon University, Silicon Valley
senaka.buthpitiya@sv.cmu.edu, ianlane@cs.cmu.edu, jike.chong@sv.cmu.edu

Goals

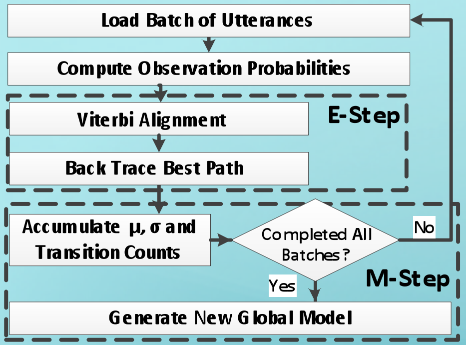
- State-of-art speech recognition systems are trained on thousands of hours of speech data, which can take many weeks even on large clusters
- Training requires:
  - Calculating observation probabilities
  - Aligning audio with transcripts
  - Estimating model parameters
  - Repeat process multiple times
- This computation bottleneck limits the number of new ideas and concepts speech experts can explore and validated in a timely manner

Carnegie Mellon University
CyLab



Approach

- Viterbi training used to estimate the parameters of an hidden-Markov-model (HMM) based acoustic model
- Leverage special left-right HMM model structure commonly used in speech recognition while heavily optimizing the observation probability computation
- Effectively organize the training algorithm into threads and thread-blocks and leverage available memory resources and synchronization capabilities to efficiently execute on a manycore computation platform



Training flow for one training iteration

Observation Probability Computation

- GMM-level parallelism · 10KB of model data · fits into scratch space on the GPU
- Threads parallelize over the observation samples
- Thread blocks parallelize over the GMMs
- Each thread in a thread block performs all computations for one time step

Alpha Computation

- Calculate optimal match between the transcript and the acoustic input
- Calculation is time-synchronous – present output depends of previous outputs
- Parallelize utterances per thread block – For optimal memory access speed

Backtracking Computation

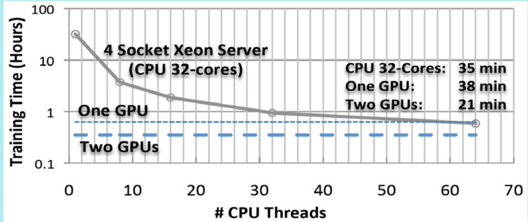
- Trace one-best path best aligning GMM states to acoustic input observations
- Naïve implementation causes severe bottleneck with excess memory reads
- We implement using a pre-fetch optimization
  - Fully utilize load bandwidth
  - Minimize memory latency caused by the pointer chasing operations

Maximization Step

- Updates aggregated statistics using aligned and labeled input observations
- Extremely large number of values to update – suffers from over/underflows
- Parallelize by mapping each utterance to a thread block
- First aggregate the histogram information within an utterance locally
- Then merge local results from each thread block to the main model

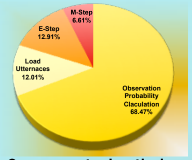
Experimental Evaluation

- GPU implementation on Intel Core i7-2600k CPU machine with two NVIDIA GTX580 GPU cards (approx. $2k)
- Traditional implementation on a 32-core Xeon server (approx. $30k)

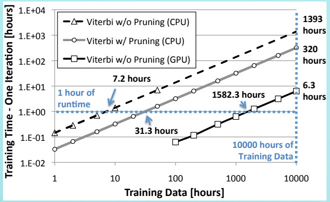


Time required for single training iteration with on a 1000hr corpus

- A 32-core Xeon server has *only* 7.5% performance advantage over a single GPU system
- With two GTX580, training **67%** faster than a 32-core Xeon server



Component-wise timing breakdown for a 1000-hour training set on (GPU)



- Speech corpora used in this evaluation consisted of 122hrs and approximately 150k utterances of speech collected from headset, lapel and far-field microphones from 168 sessions (AMI Meeting Corpus3)
- Data is replicated to generate larger training sets up to 10,000 hrs

Conclusions

1. Proposed approach is 51x faster than a sequential CPU implementation
2. Trains an acoustic model with 8000 codebook of 32-component GMs on 1000 hours of data in 9 hours
3. This empowers researchers to rapidly evaluate new ideas to build accurate and robust acoustic models on very large training corpora