An Implementation of Decision Tree-Based Context Clustering on Graphics Processing Units

Nicholas Pilkington & Heiga Zen

Cambridge University, Computer Laboratory, Cambridge, UK
Toshiba Research Europe Ltd., Cambridge Research Laboratory, Cambridge, UK

TOSHIBA
Leading Innovation

Background

- Training of HMM-based speech synthesizers consists of three parts
  1. Initialization & reestimation
  2. Embedded reestimation
  3. Decision tree-based context clustering

Training Part	Training Time (sec.)
Initialization & reestimation	**1,223**
Embedded reestimation	**9,117**
Tree-based context clustering	**40,170**

Decision tree-based context clustering takes about 80%

- Widely used implementation (HTK/HTS) is for a CPU
- Not designed for parallel architectures like GPUs

Implement decision tree-based context clustering for GPUs

Decision Tree-Based Context Clustering

- Context-dependent HMMs have been used in speech synthesis
- Phonetic, segmental, prosodic, & linguistic contexts are often used

```
pau^ao-th+er=ah@2_1
/A:0_0_0/B:1-1-2@1-2&1-7#1-4$1-3!0-2;0-4|ao/C:0+0+1
/D:0_0/E:content+2@1+5&1+2#0+3/F:in_1
/G:0_0/H:7=5@1=2|L-L%/I:7=3
/J:14+8-2
```

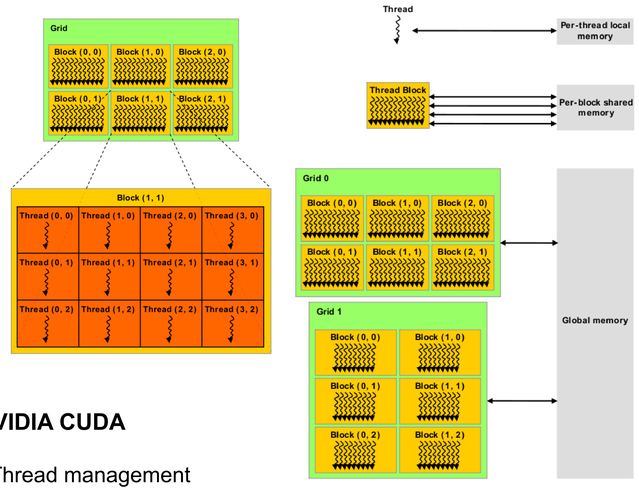
- Huge number of possible combinations of contexts
  → **Almost impossible to cover all combinations**

- Decision tree-based context clustering is used to address the problem
  - Cluster similar HMM states to sub-classes by building decision tree
  - Split is made based on their contexts by questions about contexts
  - Various criteria can be used, typically ML criterion is used
  - Tie parameters among HMM states associated with the same class

- Widely used implementation (HTK/HTS) is for a CPU
  → **Not designed for parallel architectures**

GPU Architecture

- GPUs are specialized for highly parallel computation
- Their computational power has overtaken that of CPUs
- GPU is a set of computationally powerful SIMD parallel processors
- Many-core GPUs → mainstream processor chips are now parallel



NVIDIA CUDA

- Thread management
  - Kernel
    → **Code executed on the GPU in parallel by CUDA threads**
  - CUDA threads are organized into blocks of varying dimensions
  - Each thread is given an index within its block
  - Blocks are organized into multi-dimensional grid
  - **Thread blocks are required to execute independently**
  - **Must be possible to execute in any order (parallel/series)**
    → **This allows thread blocks to be scheduled in any order across any number of cores**
  - Enables programmers to write code that scales with # of cores

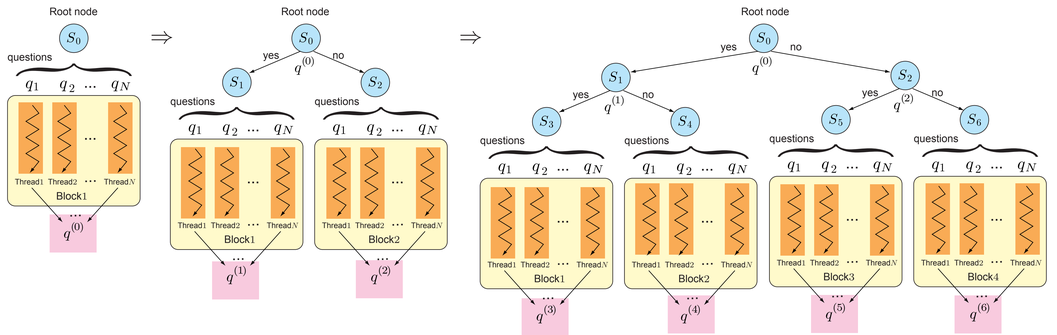
- Memory management
  - Three memory spaces
    - Private memory: accessible solely by thread
    - Shared memory: accessible to all threads in the same block
    - Global memory: accessible by all the threads at any time

Proposed Implementation of Tree-Based Context Clustering for GPUs

- Implement decision tree-based context clustering on GPUs
  → **Must be reformulated in terms of data independent sections** .
- Computing the log likelihood gain by splitting a node by a question is independent of all the other splits
  → **Can be computed concurrently with all the other questions**
- This should give large speed gain because running time is reduced
  - From: **All the possible question-cluster pairs being computed sequentially**
  - To: **All the possible question-cluster pairs being computed in parallel**

Proposed implementation of Tree-Based Clustering on GPU

0) Initially all the models are placed in a single (root) cluster
1) Launch CUDA threads, one for each pair of question & node. A CUDA thread performs
   - Accumulate statistics
   - Compute log likelihood gain
2) Wait until all threads finish
3) Select the best question to split each node
4) Split nodes by the best questions
5) Go to 1) until the stopping criterion is satisfied



Experimental Results

- Built 10 decision trees serially
- Averaged over 10 trials
- Number of distributions to be clustered was 195,273
- Number of questions about contexts was 3,238.

Architecture	Computational time (sec.)
CPU (HTS)	**47,673**
GPU (Proposed)	**4,903**
(common overheads)	**710**

GPU implementation was 11.2 times faster than HTK/HTS one

Conclusions

- Decision tree-based context clustering takes long time to build HMM-based speech synthesizers
- Proposed an implementation of tree-based clustering on GPU using NVIDIA CUDA
- **GPU implementation was 11.2 times faster than conventional CPU (HTK/HTS) implementation**