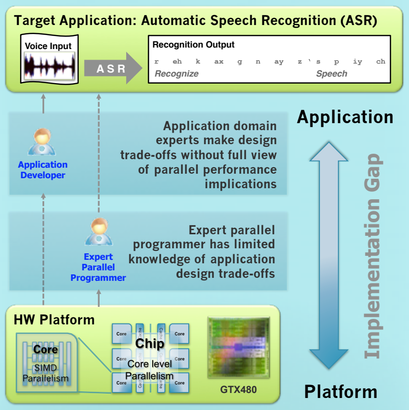
A Speech Recognition Application Framework for Highly Parallel Implementations on the GPU

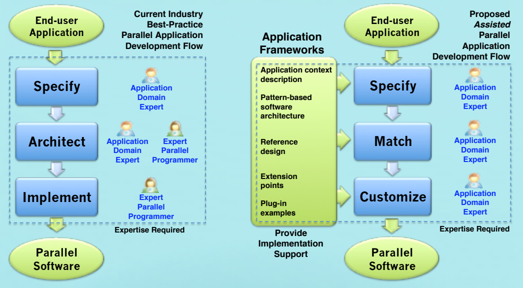
Jike Chong, Ekaterina Gonina, Kurt Keutzer, Department of Electrical Engineering and Computer Science, University of California, Berkeley
jike@eecs.berkeley.edu, egonina@eecs.berkeley.edu, keutzer@eecs.berkeley.edu
Parallel Computing Lab

The Parallel Programming Implementation Gap


Target Application: Automatic Speech Recognition (ASR)
Voice Input
ASR
Recognition Output
r eh k ax g n ay z ' s p iy ch
Recognize
Speech
Application
Application Developer
Application domain experts make design trade-offs without full view of parallel performance implications
Expert Parallel Programmer
Expert parallel programmer has limited knowledge of application design trade-offs
Implementation Gap
HW Platform
Core
SIMD Parallelism
Chip
Core level Parallelism
Core
Cache
GTX480
Platform

Both application domain expertise and parallel programming expertise are required to effectively utilize highly parallel microprocessors like the GPU

Streamlining Workflow with Guidance from an Application Framework


End-user Application
Current Industry Best-Practice Parallel Application Development Flow
Specify
Application Domain Expert
Architect
Application Domain Expert
Expert Parallel Programmer
Implement
Expert Parallel Programmer
Expertise Required
Parallel Software
Application Frameworks
Application context description
Pattern-based software architecture
Reference design
Extension points
Plug-in examples
Provide Implementation Support
End-user Application
Proposed Assisted Parallel Application Development Flow
Specify
Application Domain Expert
Match
Application Domain Expert
Customize
Application Domain Expert
Expertise Required
Parallel Software

Without the framework:

1. Specify: Highlight application characteristics
2. Architect: Define the organization of a software program in terms of parallel programming patterns
3. Implement: Construct functions, test and verify correctness and performance

Very few teams have both the application domain expertise and the parallel programming expertise

This severely limits the development and deployment on highly parallel microprocessors

With the guidance of an application framework:

1. Specify: Highlight application characteristics
2. Match: Select an application framework to use, analyze the highlighted potential bottlenecks, understand the data types and APIs
3. Customize: Leverage reference implementation and develop plug-ins for new functions

Parallel programming expertise required only in the development of the application framework

With the framework, developers with only application expertise can still benefit from GPUs

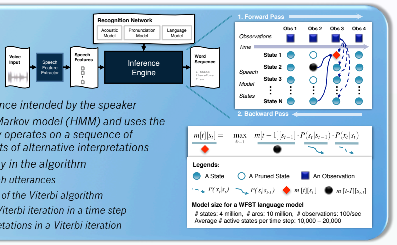
The Four Components of an Application Framework for Parallel Programming

Application Context

- Is a description of the application characteristics and requirements
- Exposes concurrency independent of the implementation platform
- For application domain experts:
  - Provides the context for understanding the motivations of parallelization decisions made in the software architecture of an application framework

Example:

- *ASR analysis of an utterance from an acoustic waveform to infer the most likely word sequence intended by the speaker*
- *Inference is based on the hidden Markov model (HMM) and uses the Viterbi algorithm, which iteratively operates on a sequence of observations and keeps track of sets of alternative interpretations*
- *There are four levels of concurrency in the algorithm*
  1. *Among different segments of speech utterances*
  2. *Among forward and backward pass of the Viterbi algorithm*
  3. *Among algorithmic steps within a Viterbi iteration in a time step*
  4. *Among different alternative interpretations in a Viterbi iteration*

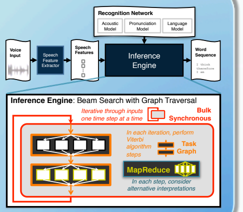


Software Architecture

- Is a hierarchical composition of parallel programming patterns that assists in navigating the reference implementation
- For application domain experts:
  - Helps to organize their efforts around the fundamental limitations and constraints of implementing the application on highly parallel microprocessors

Example:

- *The hardware targeted is the NVIDIA GTX480*
- *For efficient implementations, one must leverage the wide vector units, the GPU memory hierarchy, and the synchronization primitives within and between cores*
- *With respect to the four levels of concurrency:*
  1. *No. Data working set too large for manycore parallelism*
  2. *No. Workload not balanced, too little work in backward pass*
  3. *No. Too many intermediate operands to pass between steps*
  4. *Yes! 10,000+ way concurrency for data parallel operations, but many implementation challenges – irregular graph traversal guided by input known only at runtime, frequent memory write conflicts that require fast synchronization between cores*

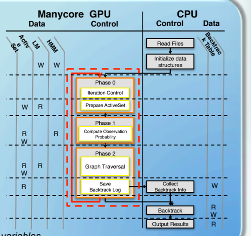

Recognition Network
Acoustic Model
Pronunciation Model
Language Model
Voice Input
Speech Feature Extractor
Speech Features
Inference Engine
Word Sequence
Inference Engine: Beam Search with Graph Traversal
Iterative through inputs one time step at a time
Bulk Synchronous
In each iteration, perform Viterbi algorithm steps
Task Graph
MapReduce
In each step, consider alternative interpretations

Reference Implementation

- Is a fully functional, efficiently implemented sample parallel design of the application
- Provides a concrete example of how each component in the application framework could be implemented, and how they can be integrated
- For application domain experts:
  - Relieves the burden of constructing functionally-correct baseline implementations before introducing new features

Example:

- *Forward pass on the GPU, backward pass on the CPU*
- *Challenges resolved in the forward pass on the GPU:*
  1. *Constructed efficient dynamic vector data structures to handle irregular graph traversals*
  2. *Implemented an efficient find-unique function to eliminate redundant work by leveraging the GPU global memory write-conflict-resolution policy*
  3. *Implemented lock-free accesses of a shared map leveraging advanced GPU atomic operations to enable conflict-free reduction*
  4. *Used hybrid local/global atomic operations and local buffers for the construction of a global queue to avoid sequential bottlenecks in accessing global queue control variables*

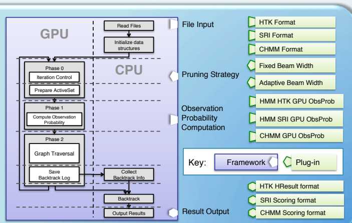


Extension Points

- Are a set of interfaces defined to summarize the interactions between the application framework and potential new modules
- For application domain experts:
  - Provide flexible interfaces for implementing plug-ins to extend the framework functions without jeopardizing the execution efficiency in the application framework

Example:

- *Extension points implemented using* Abstract Factory *creational object-oriented programming pattern*
- *Three extension points implemented:*
  1. ***Observation Probability Computation***
  2. ***Pruning Strategy***
  3. ***Result Output***
- *Many pre-defined plug-ins available*
- *New plug-ins can be developed by application domain experts*



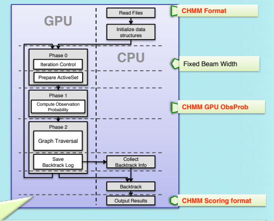
Thanks to Dorothea Kolossa, Steffen Zeiler for their collaboration in the case study.
Thanks to Nelson Morgan, Andreas Stolcke, and Adam Janin at ICSI for insightful discussions and continued support in the infrastructure used in this research.
This research is supported in part by an Intel Ph.D. Fellowship.
This research also supported in part by Microsoft (Award #024263) and Intel (Award #024894) funding and by matching funding by U.C. Discovery (Award #DIG07-10227).

Want to learn more about this topic?

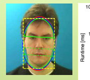
Session 2046 - Efficient Automatic Speech Recognition on the GPU
Thursday, September, 23rd, 15:00 - 15:50

Case Study with an Application Domain Expert ➔ 20x Application Performance Improvement on GPU

- Robustness of speech recognition can be significantly improved by multi-stream inputs and especially by **audio-visual speech recognition** (enabling lip-reading)
- Coupled hidden Markov models (CHMMs), with their tolerance for stream asynchronicities, can provide a flexible integration of these streams
- Targets human-computer interactions in noisy reverberant environments:
  - Ticket machine in train stations, information booth in tourist hot spots
- Using the ASR application framework:
  - A CHMM can be compiled into a WFST for use as speech model
  - Observation Probability Computation was extended with a new plug-in to handle multiple streams
  - New input/output plug-ins
- Platforms used:
  - CPU: i7 920, 12GB mem, sequential application using one of the four cores
  - GPU: GTX480, 1.5GB mem, data parallel operation on 15 multiprocessor cores



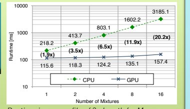
Prof. Dorothea Kolossa
Speech Application Domain Expert
Technische Universität Berlin



Extended *audio-only speech recognition* framework to enable *audio-visual speech recognition* (***lip reading***)

Achieved a ***20x speedup*** in application performance compared to a sequential version in C++

The application framework enabled a ***Matlab/Java programmer*** to ***effectively utilize a highly parallel platform***



Runtime in ms per file of 3s length for M = 1,2,...,16 mixture components. The speedup factor is given in parentheses.

Dorothea Kolossa, Jike Chong, Steffen Zeiler, Kurt Keutzer, "Efficient Manycore CHMM Speech Recognition for Audiovisual and Multistream Data", To be published at Interspeech 2010.

Key Lessons

- Application framework for parallel programming is developed to help application domain experts effectively utilize highly parallel Microprocessors
- The ASR application framework has enabled a Matlab/Java programmer to achieve 20x speedup in her application by extending an audio-only speech recognition reference implementation to an audio-visual speech recognition application
- It is an effective approach for transferring tacit knowledge about efficient, highly parallel software design for use by application domain experts
- With the proliferation of highly parallel computation from servers to workstations to laptops and portable devices, there will be an increasing demand for adapting business and consumer applications to specific usage scenarios
- Application frameworks for parallel programming will be an important force for accelerating the adoption of highly parallel microprocessors