Scalable Large-Vocabulary Continuous Speech Recognition

Katya Gonina
with Jike Chong, Kisun You, Youngmin Yi, Kurt Keutzer & others

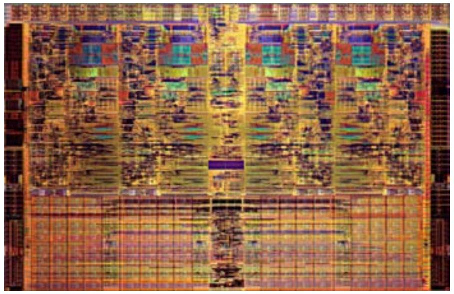
UC Berkeley ParLab

January 30, 2012

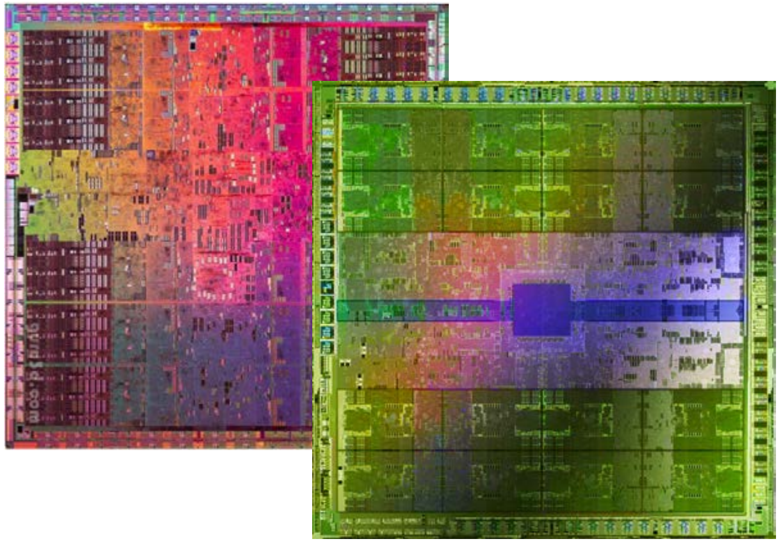
Scalability

Parallel scalability:

The ability for an application to efficiently utilize an increasing number of processing elements



Intel Core i7 (45nm)
4 cores



NVIDIA GTX280 and GTX480
30 and 14 cores

Parallel scalability is required for software to obtain sustained performance improvements on successive generations of processors

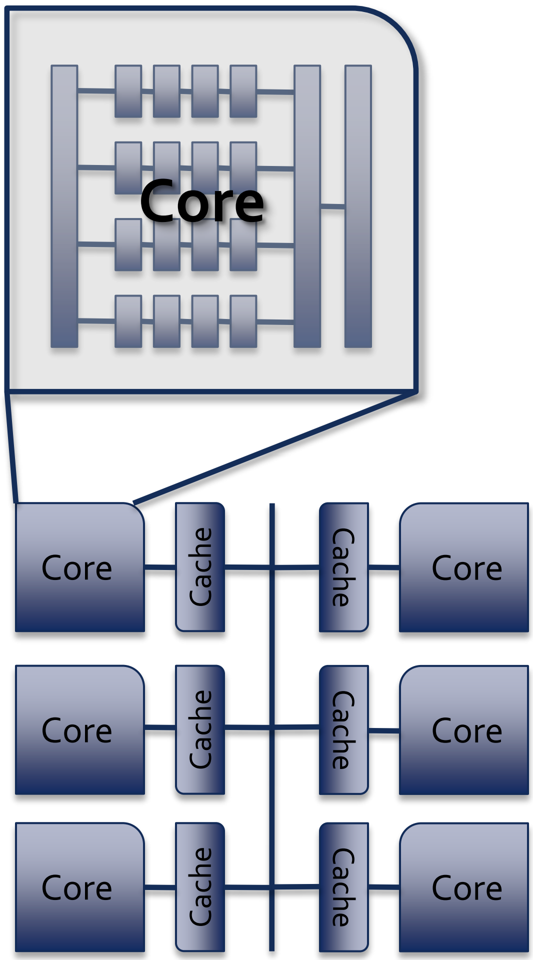
Outline

- Characteristics of Manycore Architectures
- Speech Recognition Application
  - Software architecture and characteristics
  - Important parallelization concerns
  - Design space explored for application scalability
- Design Space Evaluation
- Recognition Network Structure Evaluation
- Conclusion

Outline

- **Characteristics of Manycore Architectures**
- Speech Recognition Application
  - Software architecture and characteristics
  - Important parallelization concerns
  - Design space explored for application scalability
- Design Space Evaluation
- Recognition Network Structure Evaluation
- Conclusion

Parallel Platform Characteristics



- Multicore/manycore design philosophy
  - **Multicore:** Devote significant transistor resources to single thread performance
  - **Manycore:** Maximizing computation throughput at the expense of single thread performance
- Architecture Trend:
  - Increasing vector unit width (SIMD)
  - Increasing numbers of cores per die
- Application Implications:
  - Must increase data access regularity
  - Must optimize synchronization cost

We explore a ***design space*** for ***application scalability*** for a speech inference engine on multicore and manycore platforms

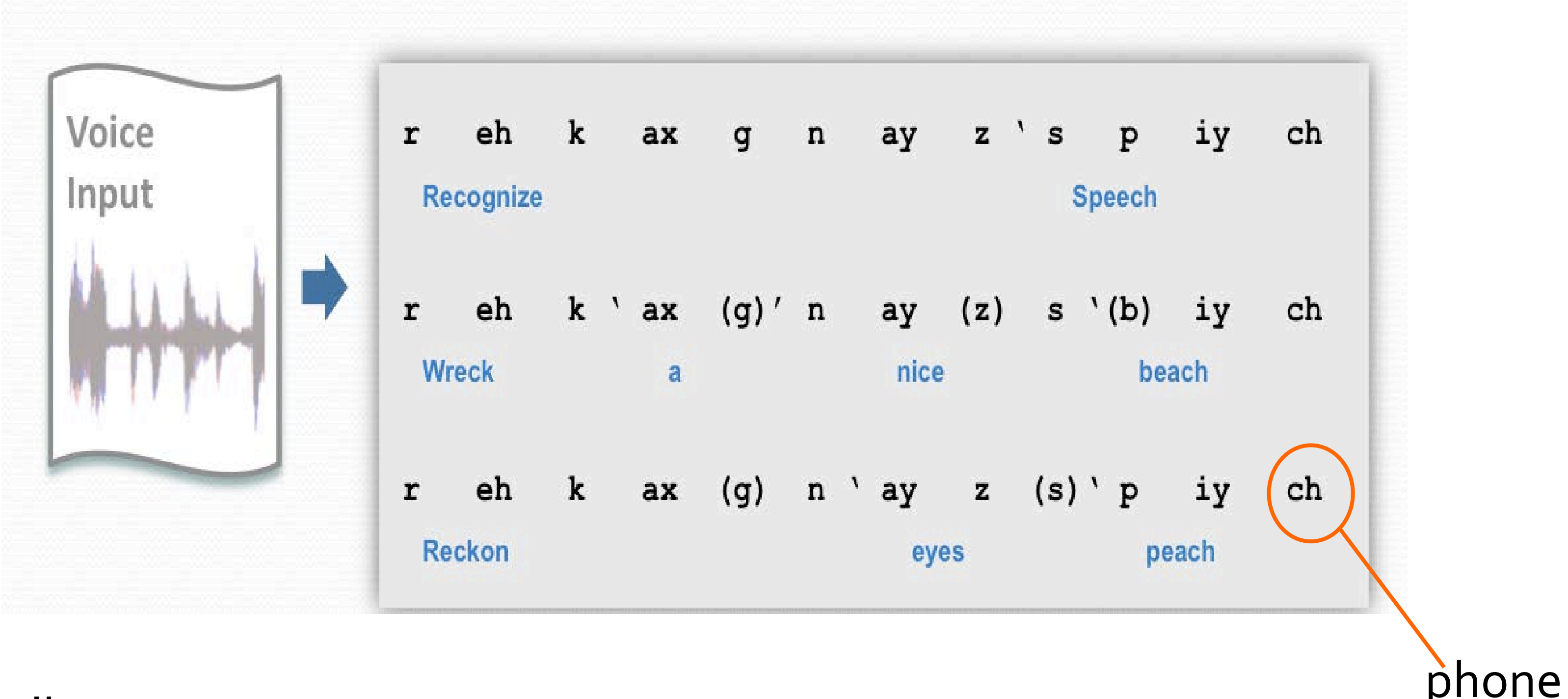
Outline

- Characteristics of Manycore Architectures
- **Speech Recognition Application**
  - Software architecture and characteristics
  - Important parallelization concerns
  - Design space explored for application scalability
- Design Space Evaluation
- Recognition Network Structure Evaluation
- Conclusion

Outline

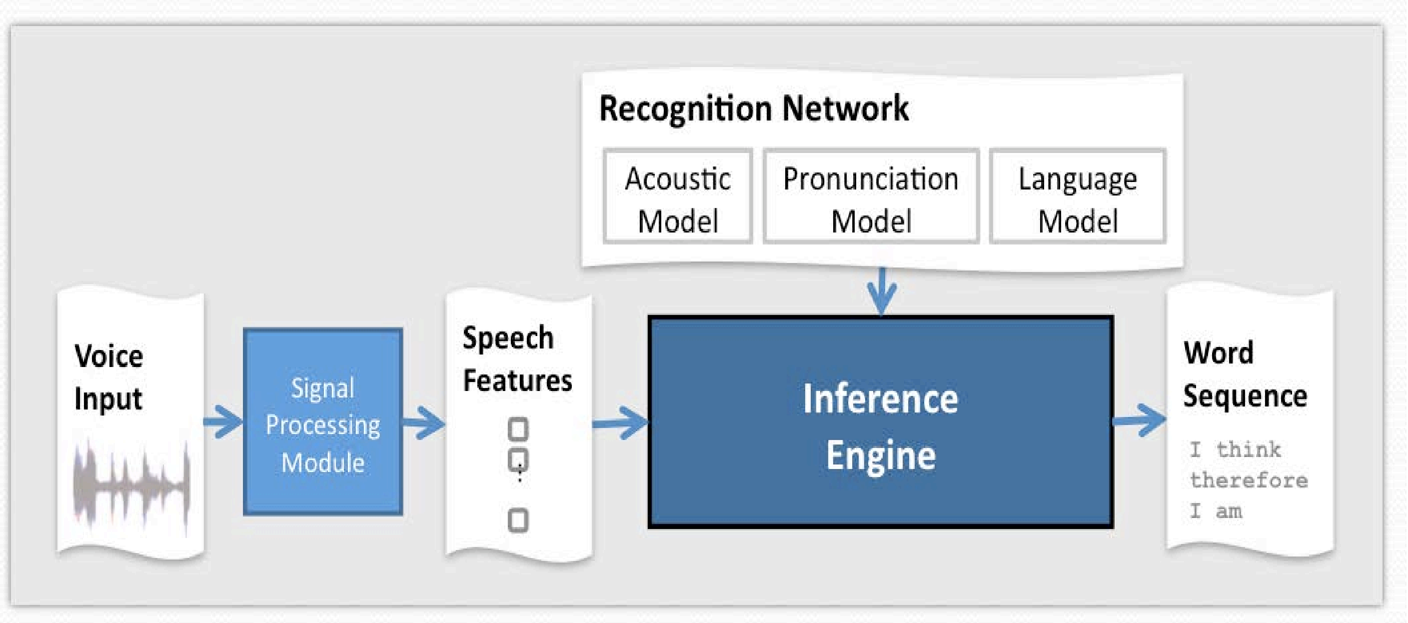
- Characteristics of Manycore Architectures
- **Speech Recognition Application**
  - **Software architecture and characteristics**
  - **Important parallelization concerns**
  - Design space explored for application scalability
- Design Space Evaluation
- Recognition Network Structure Evaluation
- Conclusion

Continuous Speech Recognition



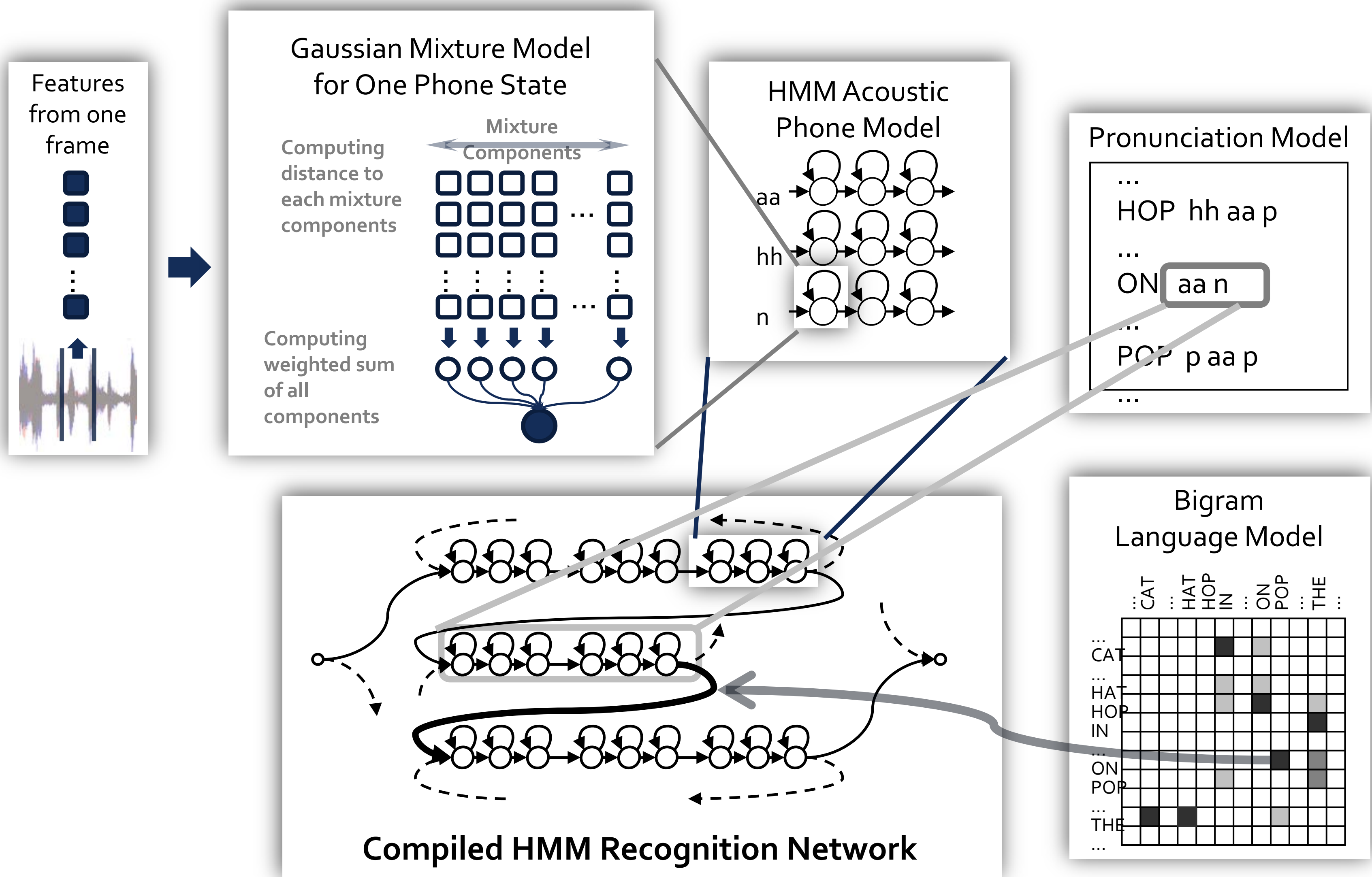
- Challenges:
  - Recognizing words from a large vocabulary arranged in exponentially many possible permutations
  - Inferring word boundaries from the context of neighboring words
- Viterbi algorithm on Hidden Markov Models (HMM) is currently the most popular approach

Continuous Speech Recognition

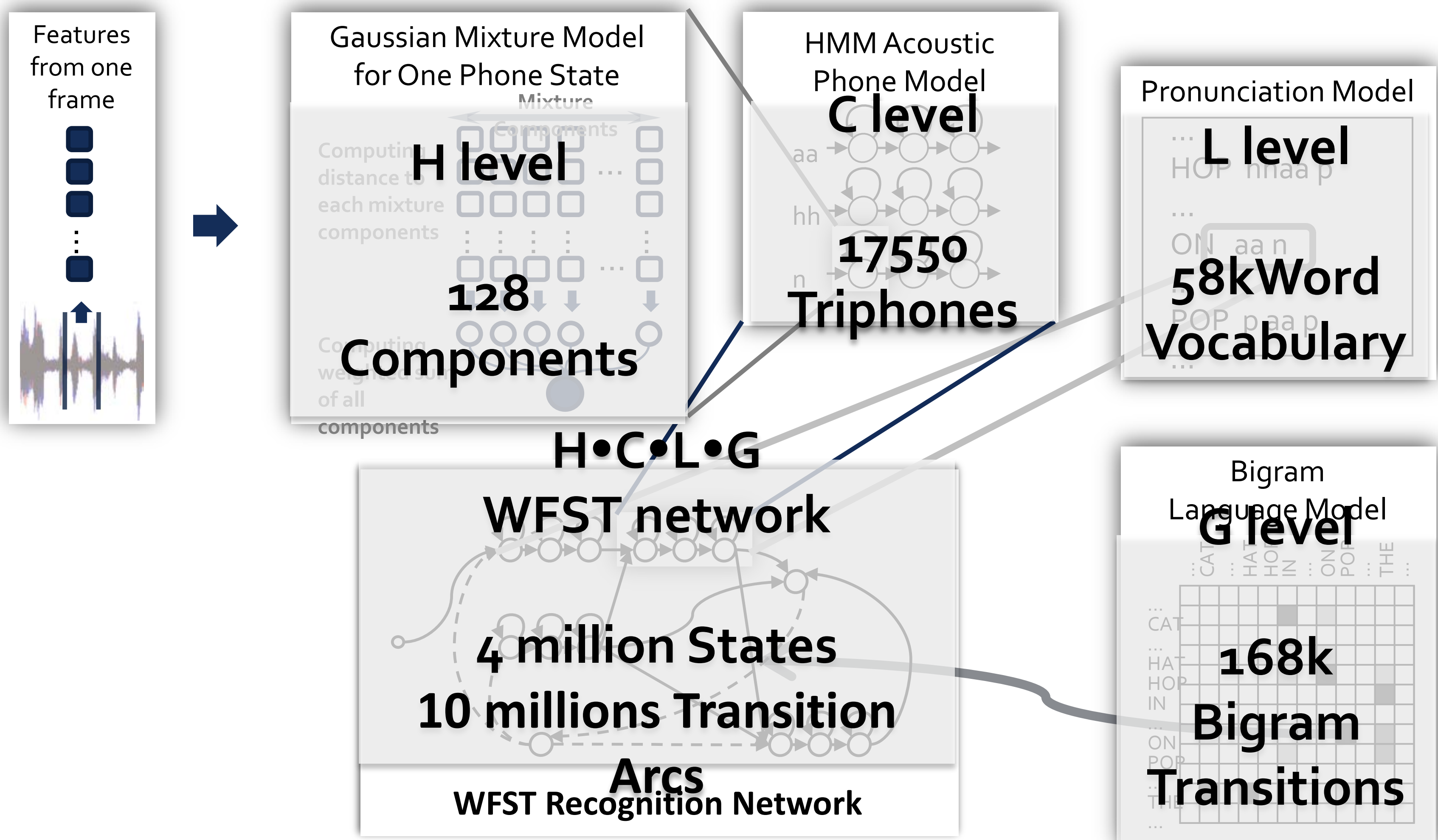


- Inference engine system
  - Used in Sphinx (CMU, USA), HTK (Cambridge, UK), and Julius (CSRC, Japan)
- Modular and flexible setup
  - Shown to be effective for Arabic, English, Japanese, and Mandarin

Recognition Network

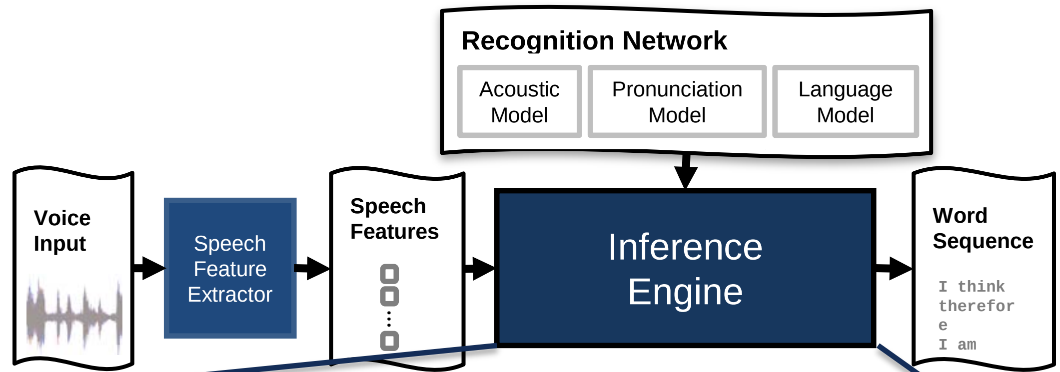


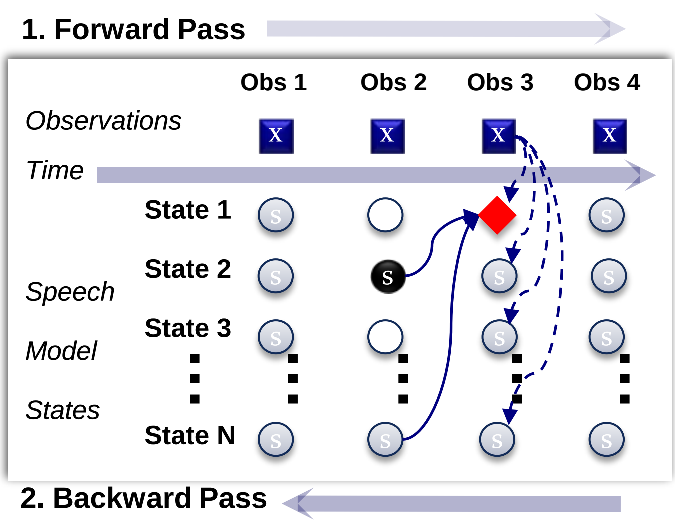
Recognition Network



Speech Inference: Detailed Algorithm

HMM-based inference





$$m[t][s_t] = \max_{s_{t-1}} m[t-1][s_{t-1}] \cdot P(s_t|s_{t-1}) \cdot P(x_t|s_t)$$

Legends:

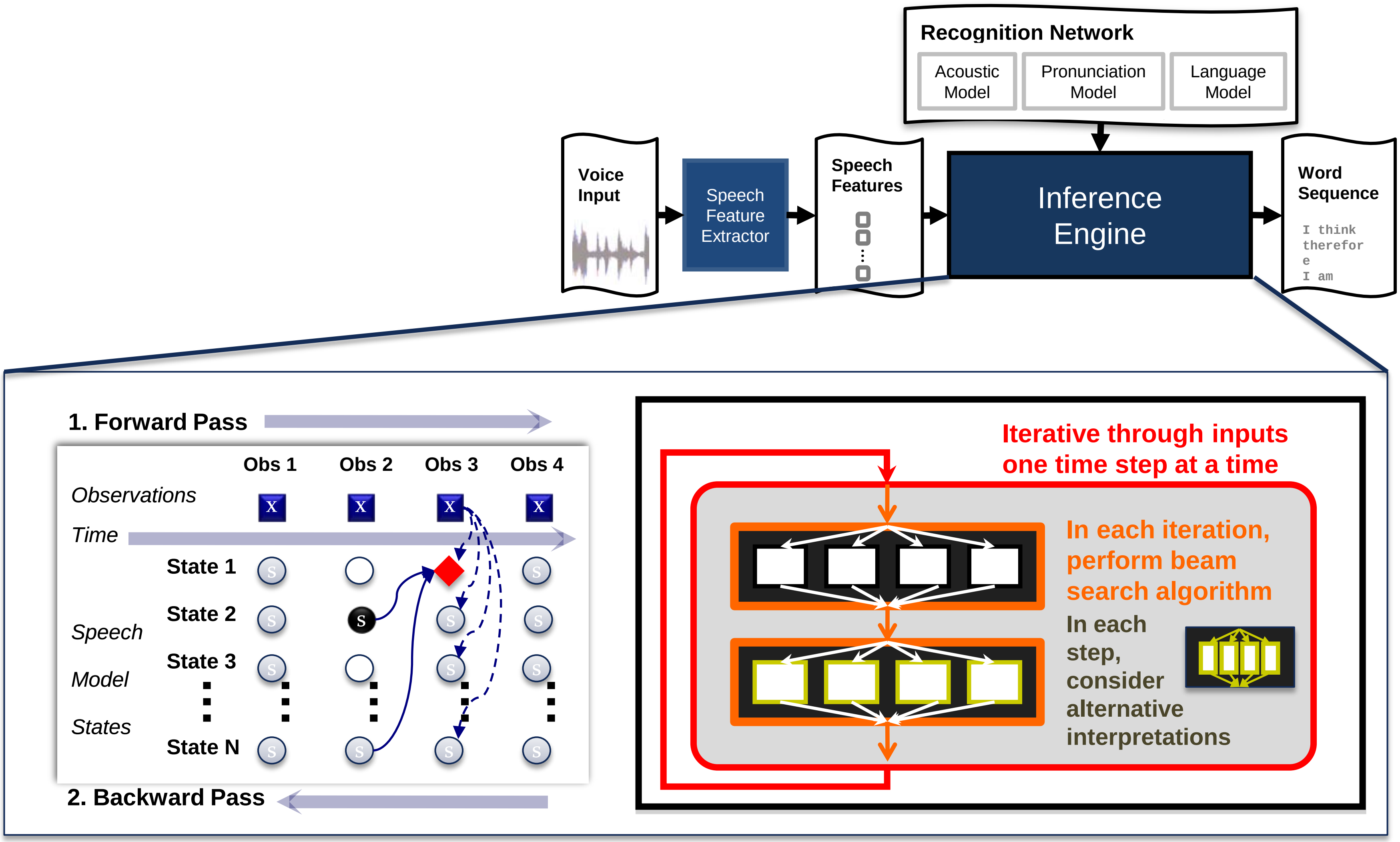
A State   A Pruned State   An Observation

$P(x_t|s_t)$ $P(s_t|s_{t-1})$ $m[t][s_t]$ $m[t-1][s_{t-1}]$

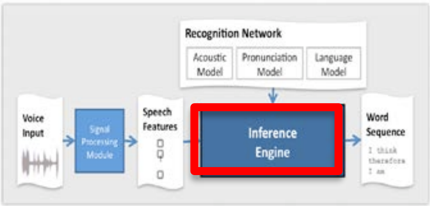
Model size for a WFST language model

states: 4 million, # arcs: 10 million, # observations: 100/sec
Average # active states per time step: 10,000 – 20,000

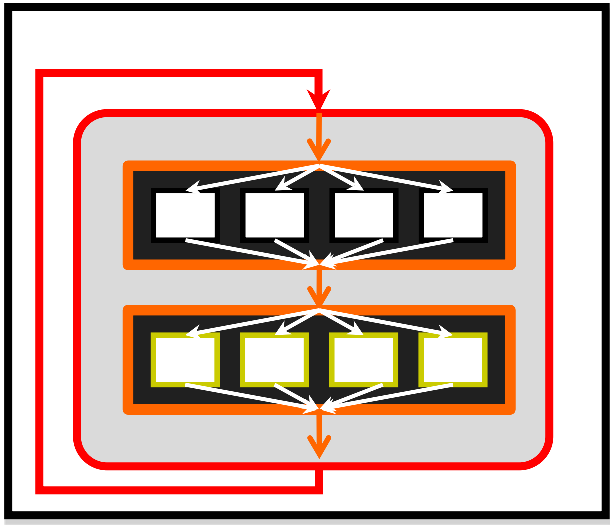
ASR: Detailed Algorithm

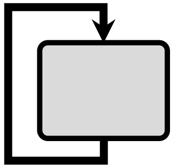


Inference Engine Architecture



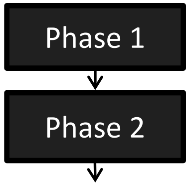
- A highly hierarchical structure
  - An iterative outer loop over time steps
  - A pipeline of operations in each time step
  - A set of alternative hypothesis to advance





One iter per time step: (~60M inst)

Sequential operation with iteration dependencies

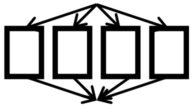


Obs prob compute

Graph traversal

Compute Intensive

Communication Intensive

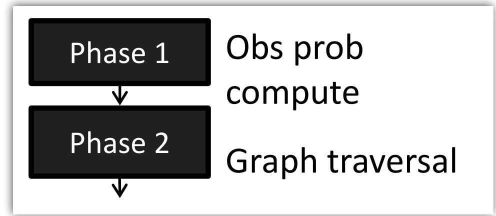


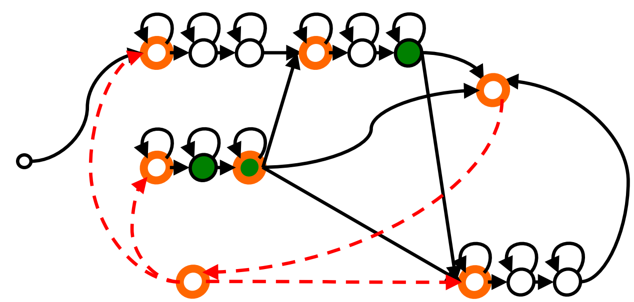
Multiple steps in a phase, each has: 1000s to 10,000s concurrent tasks (10 to 500 instr.)

Extensive fine-grained parallelism at the inner most level

Recognition Process

- **Phase 1:**
  - Observation probability computation
  - Highly compute intensive step
- **Phase 2:**
  - Traverse out-going arcs from active states
  - Write contention must be resolved at the destination states
  - Destination state is updated with most-likely in-coming arc





WFST Recognition Network

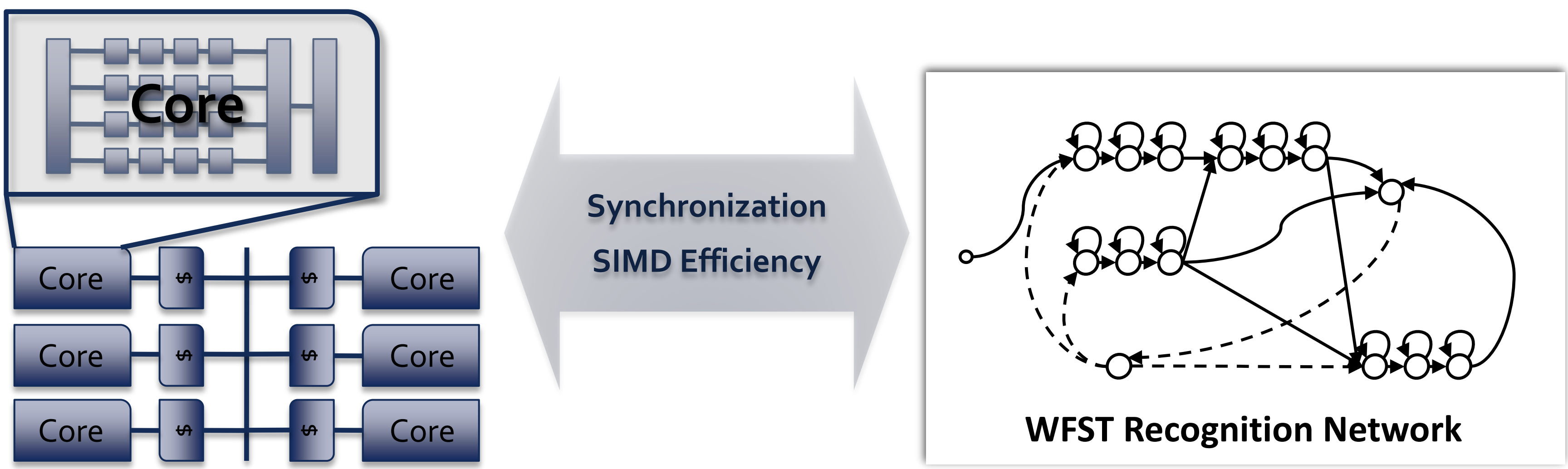
Recognition is a process of graph traversal

Inference Engine Challenges

- Application Challenges
  - Irregularity of network
  - Input-dependent, dynamically changing working set
- Scalability Goals
  - Expose sufficient concurrency

    1) Efficiently synchronize between an increasing number of concurrent tasks

    2) Effectively utilize all levels of parallel resources, including SIMD parallelism

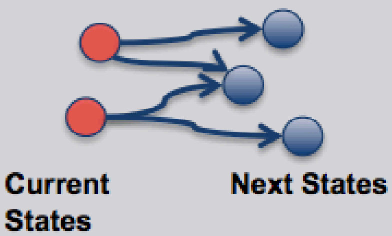
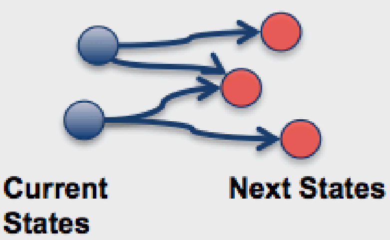


Outline

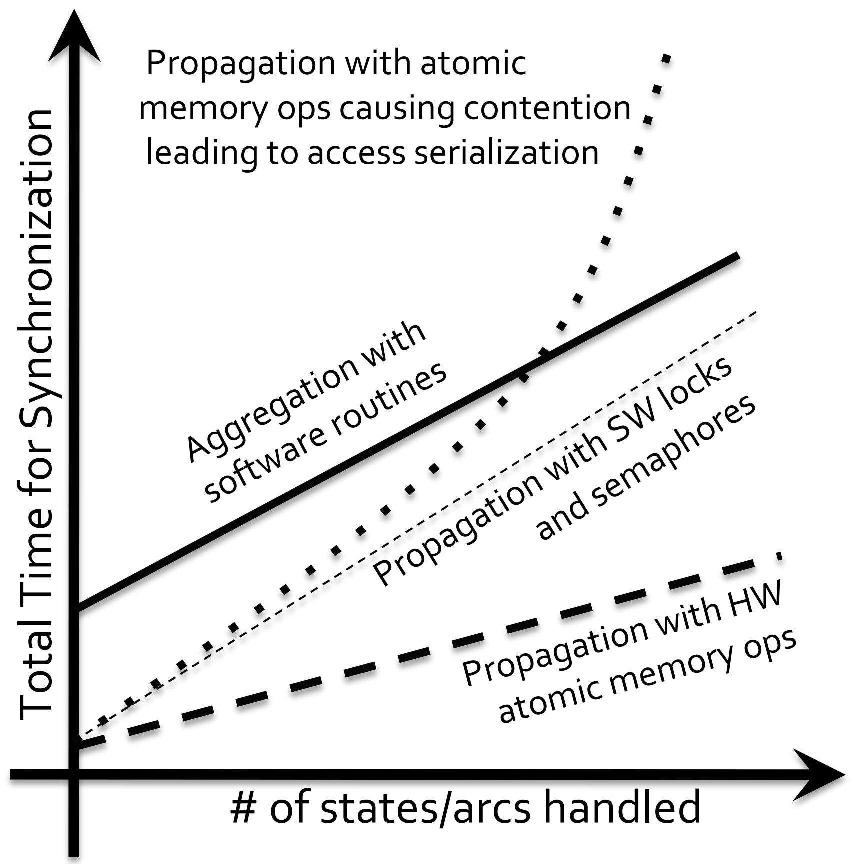
- Characteristics of Manycore Architectures
- **Speech Recognition Application**
  - Software architecture and characteristics
  - Important parallelization concerns
  - **Design space explored for application scalability**
- Design Space Evaluation
- Recognition Network Structure Evaluation
- Conclusion

Core Level Synchronization

- Challenge:
  - The cost for write conflict resolution can dominate runtime
- Experiment:
  - Allow traversal to either **propagate** from source or **aggregate** at destination for write conflict resolution

	Advantages	Disadvantages	Figure
Traversal by Propagation	Easy to program, HW handles write conflicts transparently	Sensitive to atomic operation latency	
Traversal by Aggregation	Explicit resolution of write conflicts, no atomics	Overhead in building lists of to-be-updated destination states	

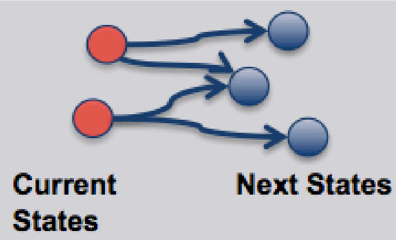
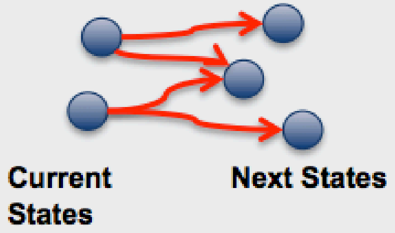
Synchronization Cost



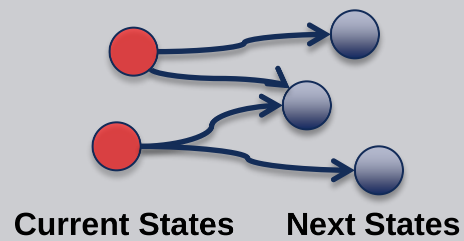
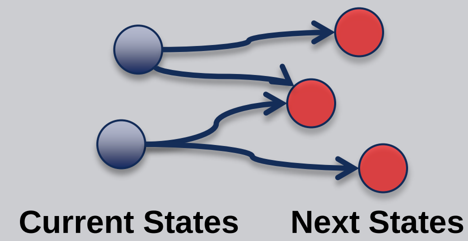
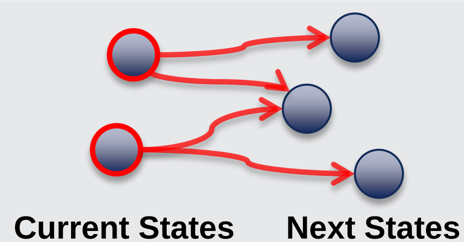
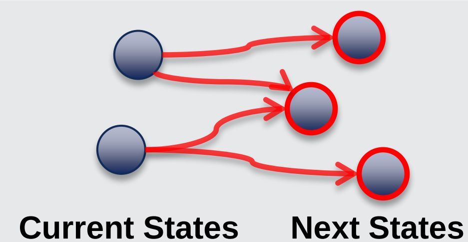
- The fixed cost (overhead) of ***aggregation*** technique is significant
- Relative gradient of ***propagation*** and ***aggregation*** techniques depend on the efficiency of the platform in resolving write conflicts
- If no hardware atomics are available, using spin locks and semaphores will be costly
- If data structure requires multiple writes to the same destination states, significant contention can occur

SIMD Utilization Efficiency

- Challenge:
  - Vector unit efficiency can quickly drop off with increased vector width
- Experiment:
  - Traverse the recognition network based on **active states** or **active arcs**

	Advantages	Disadvantages	Figure
Active States	Easy to program, all active arcs emit from active states	Load-imbalance, number of arcs varies per state	
Active Arcs	Finer granularity, Load balance	More information to maintain more arcs than states	

Design Space

	Traversal by Propagation	Traversal by Aggregation
Active States	 Maintain active source states, propagate out-arc computation results to destination state	 Maintain active destination states, determine all potential destination states and aggregate incoming arcs
Active Arcs	 Maintain active arcs, propagate active arc computation results to destination state	 Maintain active arcs, group arcs with same destination states and aggregate active arcs locally to resolve write conflicts

Hardware Platform

Specifications	Core i7920	GTX280
Processing Elements	4 cores (SMT), 4 way SIMD @2.66 GHz	30 cores, 8 way physical, 32 way logical SIMD @1.3 GHz
SP GFLOP/s	85.1	933
Memory Bandwidth	25.6 GB/s	141 GB/s
Register File	-	1.875 MB
Local Store	-	480 kB

Outline

- Characteristics of Manycore Architectures
- Speech Recognition Application
  - Software architecture and characteristics
  - Important parallelization concerns
  - Design space explored for application scalability
- **Design Space Evaluation**
- Recognition Network Structure Evaluation
- Conclusion

Efficiency vs Platform

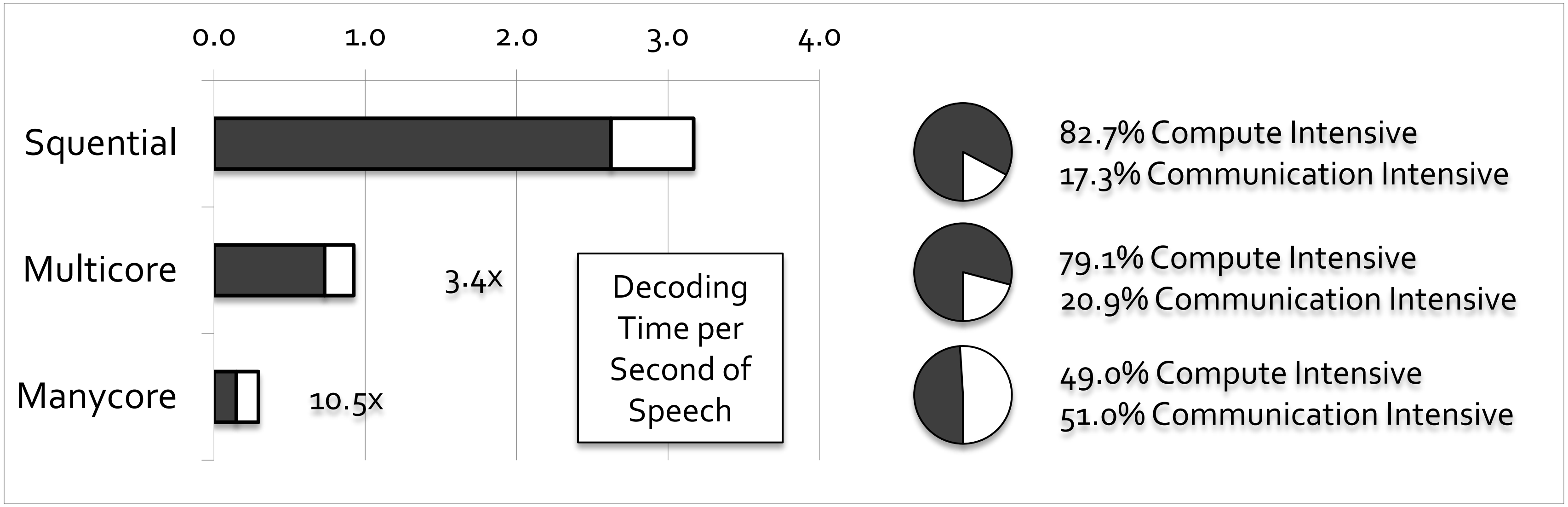
[TABLE 3] RECOGNITION PERFORMANCE NORMALIZED FOR 1 S OF SPEECH FOR DIFFERENT ALGORITHM STYLES. SPEEDUP REPORTED OVER OPTIMIZED SEQUENTIAL VERSION OF THE PROPAGATION-BY-STATES STYLE.

	CORE i7	CORE i7			GTX280			
SECONDS (%)	SEQUENTIAL PROP. BY STATES	PROP. BY STATES	PROP. BY ARCS	AGGR. BY STATES	PROP. BY STATES	PROP. BY ARCS	AGGR. BY STATES	AGGR. BY ARCS
PHASE 1	2.623 (83%)	**0.732 (79%)**	0.737 (73%)	0.754 (29%)	0.148 (19%)	**0.148 (49%)**	0.147 (12%)	0.148 (16%)
PHASE 2	0.474 (15%)	**0.157 (17%)**	0.242 (24%)	1.356 (52%)	0.512 (66%)	**0.103 (34%)**	0.770 (64%)	0.469 (51%)
PHASE 3	0.073 (2%)	**0.035 (4%)**	0.026 (3%)	0.482 (19%)	0.108 (15%)	**0.043 (14%)**	0.272 (23%)	0.281 (31%)
SEQUENTIAL OVERHEAD	—	**0.001**	0.001	0.001	0.008 (1.0%)	**0.008 (2.5%)**	0.014 (1.2%)	0.014 (1.6%)
TOTAL	3.171	**0.925**	1.007	2.593	0.776	**0.301**	1.203	0.912
SPEEDUP	1	**3.43**	3.15	1.22	4.08	**10.53**	2.64	3.48

Recognition Accuracy

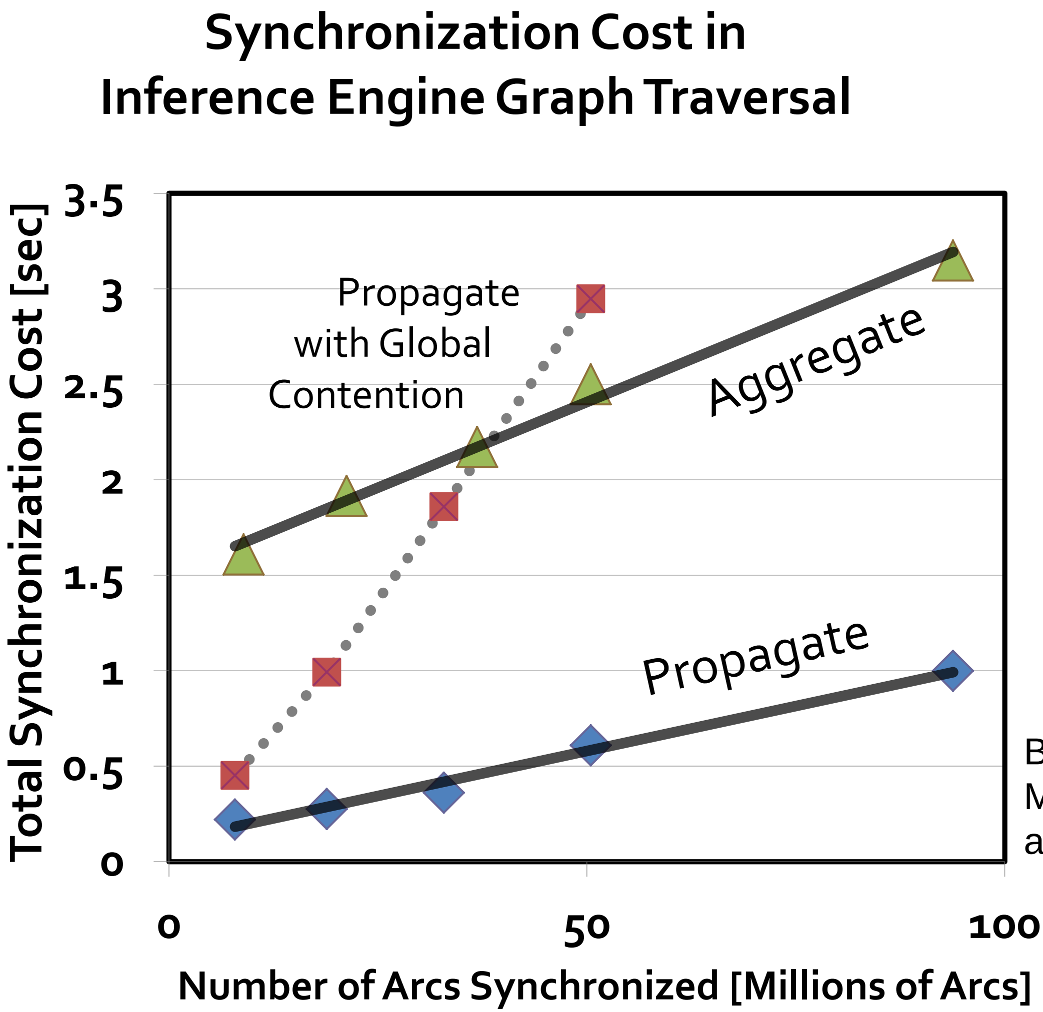
Avg. # of Active States		32820	20000	10139	3518
Word Error Rate		41.6	41.8	42.2	44.5
RTF	Sequential	4.36	3.17	2.29	**1.2**
	Multicore	**1.23**	0.93	0.70	**0.39**
	Manycore	**0.40**	0.30	0.23	0.18

Overall Speedup



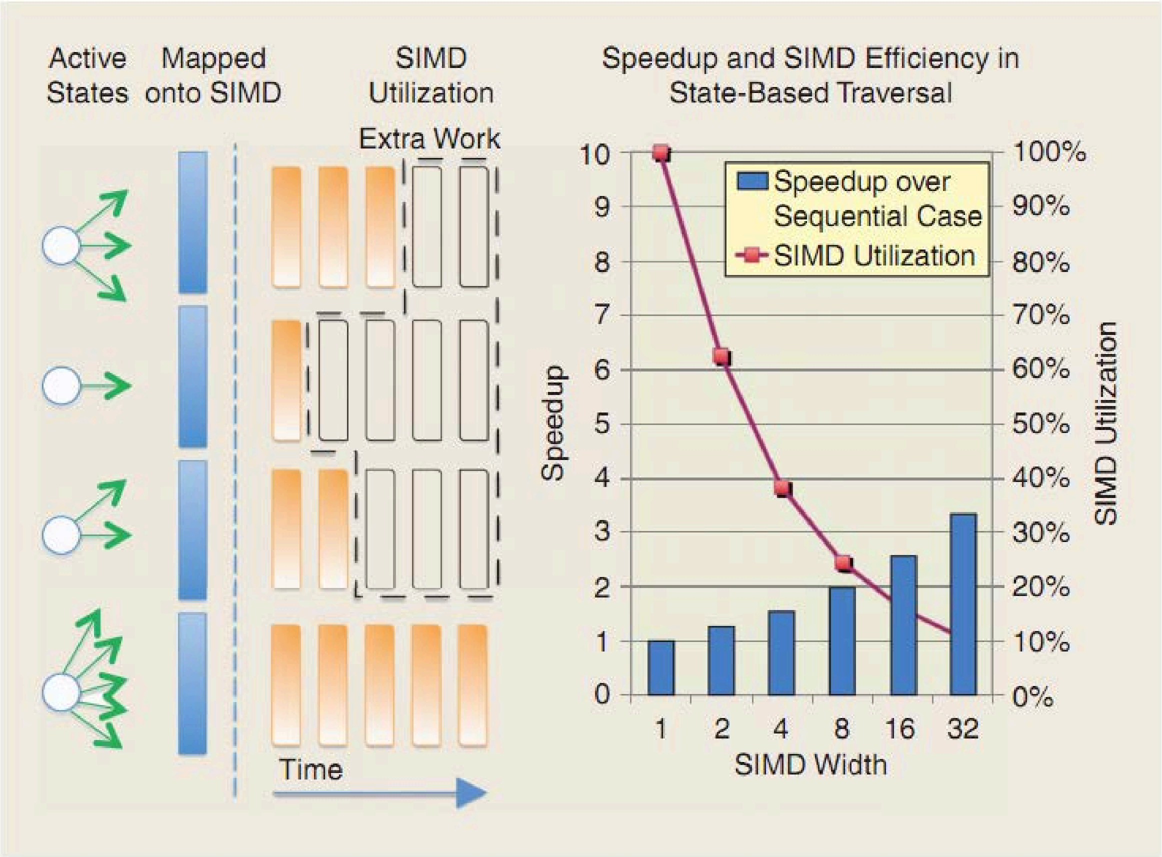
- Speed up varies between phases
  - 4-20x for compute intensive phases
  - 3-4x for communication intensive phases
  - Communication intensive phases becoming proportionally more important

Synchronization Cost



Based on Manycore analysis

SIMD Utilization Efficiency



	State Based	Arc Based
Time taken	756.79 ms	81.74 ms
Speedup	1x	9.25x

Outline

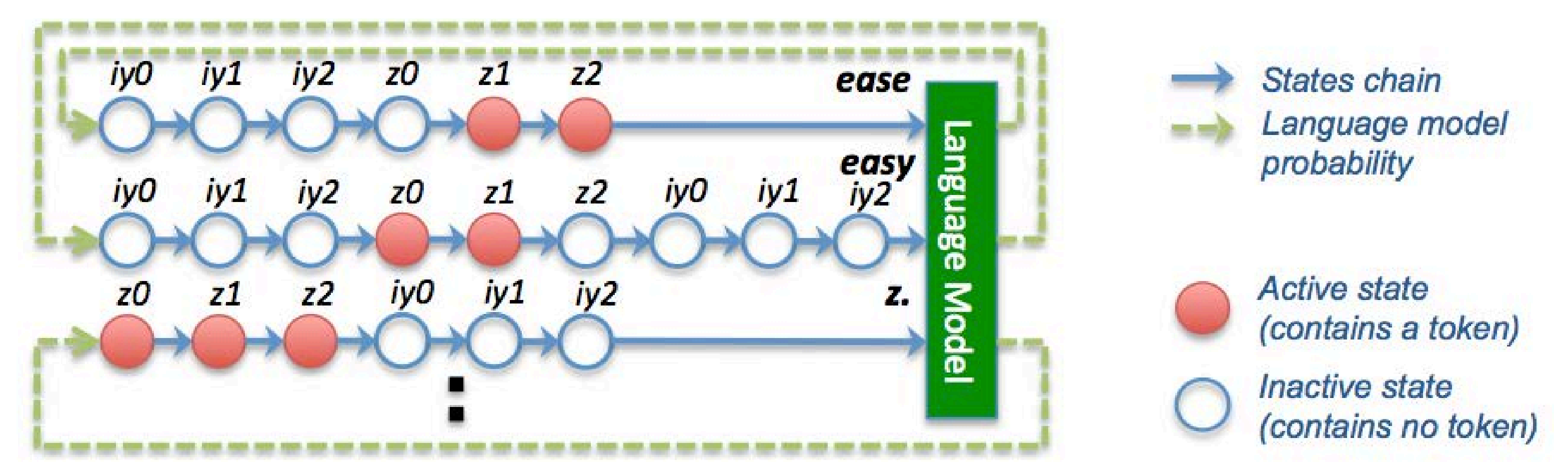
- Characteristics of Manycore Architectures
- Speech Recognition Application
  - Software architecture and characteristics
  - Important parallelization concerns
  - Design space explored for application scalability
- Design Space Evaluation
- **Recognition Network Structure Evaluation**
- Conclusion

Recognition Network Representation

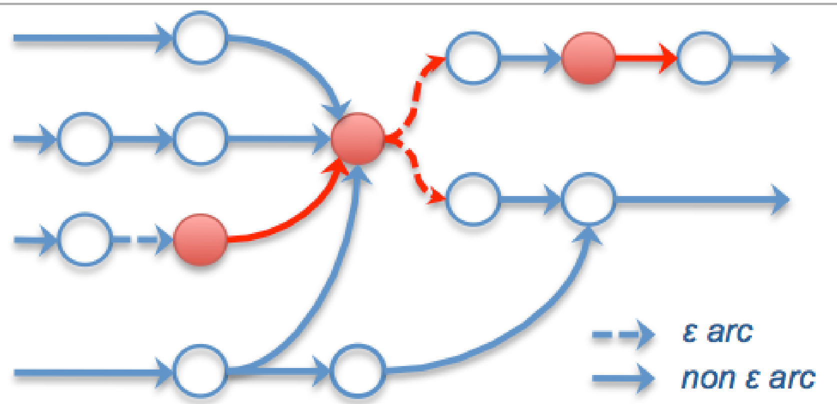
- Significant effort put into optimizing recognition networks
- Starting at baseline Linear Lexical Models
  - One chain of states per word
- Tree-lexical
- Finite state machine techniques to construct WFST

What implications does the structure have on efficiency of parallel speech inference algorithms?

Linear-Lexical Model vs WFST



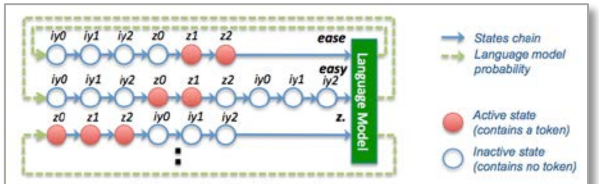
A sample Linear Lexical Model Network



A section of a Weighed Finite State Transducer Network

Inference Implementation Using LLM Network

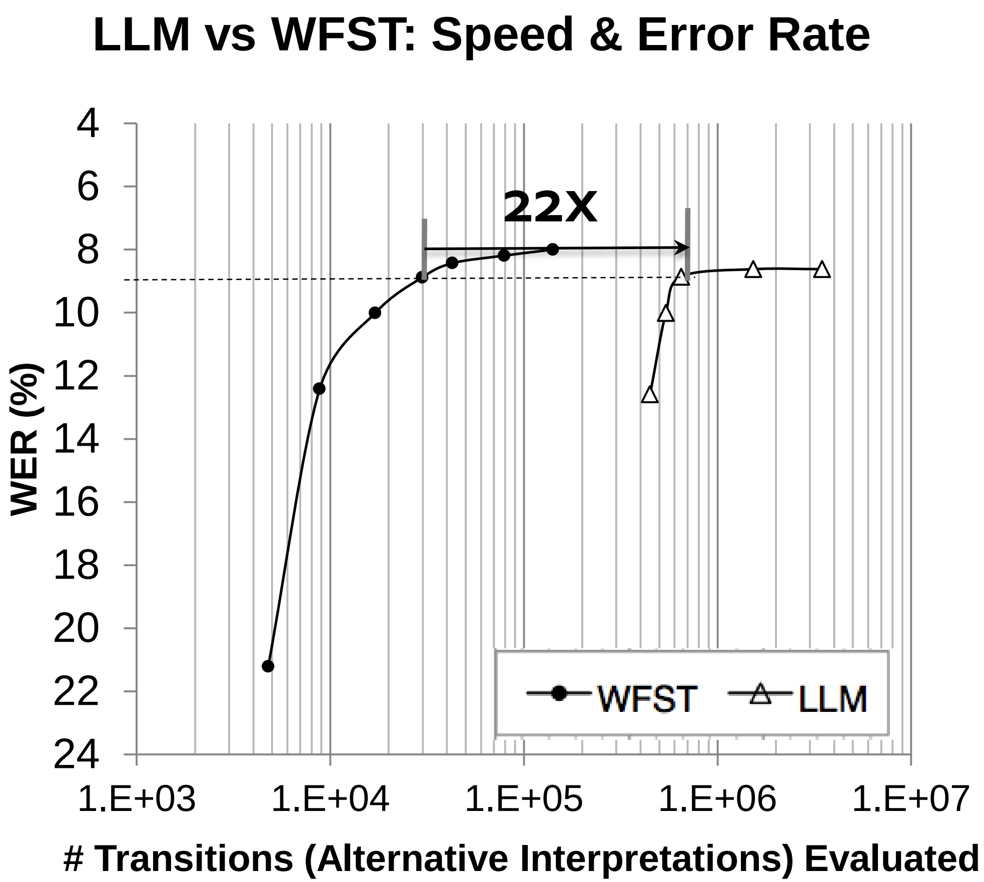
- Explicitly handles two types of transitions
  - Within-word
  - Across-word
- Optimized data layout for each type
  - First states for each word stored consecutive for across-word transitions
  - Chains of within-word states stored as a chain
- Across-word transitions – all-to-all dense computation
  - Extremely efficient on the GPU



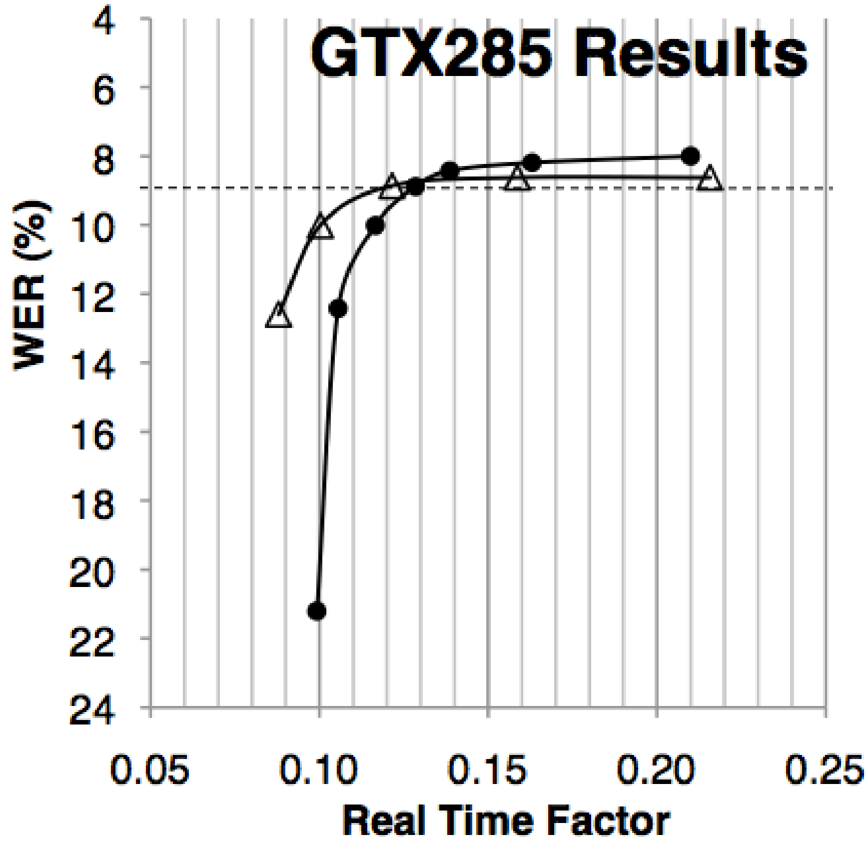
A sample Linear Lexical Model Network

Results

- Wall Street Journal 5K Corpus

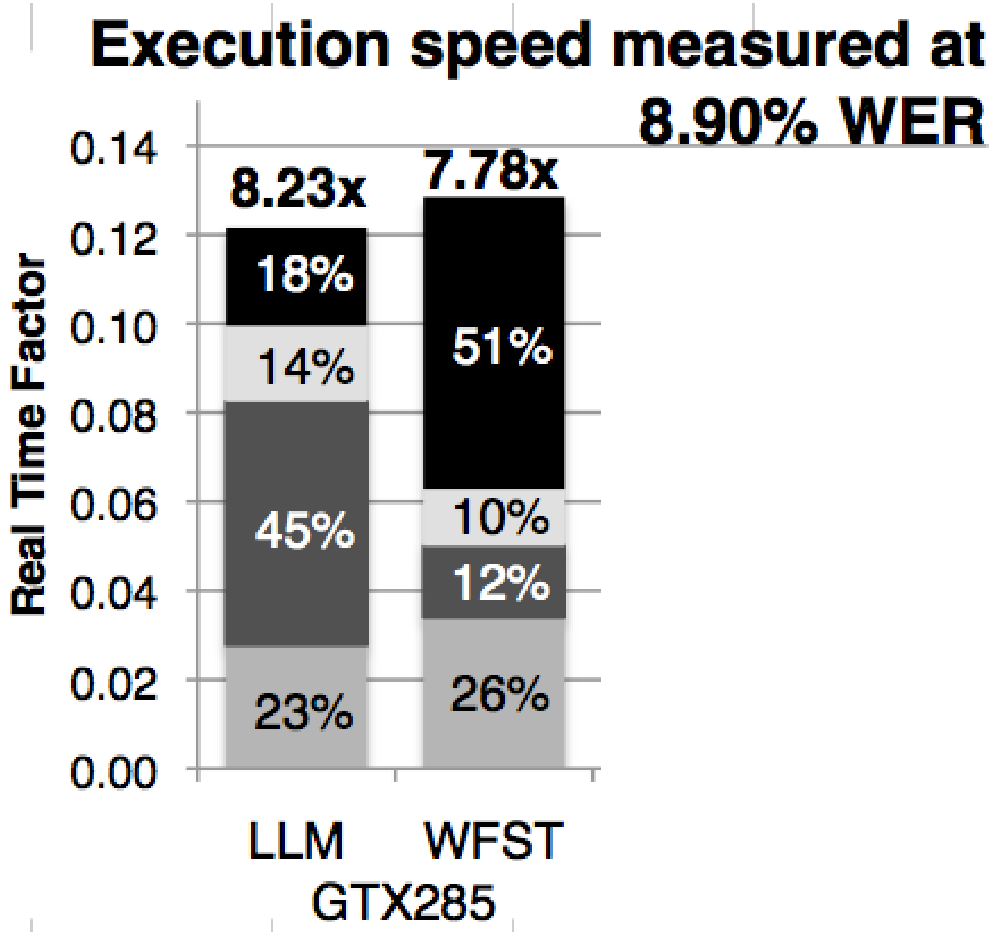


Results



GTX285 Results
WER (%)
4
6
8
10
12
14
16
18
20
22
24
0.05
0.10
0.15
0.20
0.25
Real Time Factor

Execution Time



Data Gathering | Observation Prob | Graph Traversal | Sequential Overhead

Outline

- Characteristics of Manycore Architectures
- Speech Recognition Application
  - Software architecture and characteristics
  - Important parallelization concerns
  - Design space explored for application scalability
- Design Space Evaluation
- Recognition Network Structure Evaluation
- **Conclusion**

Conclusions

- Scalable software architecture for speech recognition inference engine
  - 2.5% sequential overhead
- Explored algorithmic design space
  - Fastest algorithm depends on platform
  - Core synchronization and SIMD optimization are important for scalability
- Explored recognition network representation
  - Simpler, more regular LLM representation very competitive with highly-optimized, more irregular WFST

Current and Future Work

- Efficient training of acoustic models (GMMs)
- Productive parallel computing for application writers
  - Not have to go through this process every time
- Automating parallelization techniques
  - High-level code transformation
  - Just-in-time compilation
  - Code variant selection
- What is the best (parallel) platform for a particular algorithm?

Thank you!

Research supported by Microsoft (Award #024263) and Intel (Award #024894) funding and by matching funding by U.C. Discovery (Award #DIG07-10227). Additional support comes from Par Lab affiliates National Instruments, Nokia, NVIDIA, Oracle, and Samsung.

References

[1] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick, "The landscape of parallel computing research: A view from Berkeley," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2006-183, Dec 2006.

[2] A. Obukhov and A. Kharlamov, "Discrete cosine transform for 8x8 blocks with CUDA," NVIDIA white paper, October 2008.

[3] V. Podlozhnyuk, "FFT-based 2D convolution," NVIDIA white paper, June 2007.

[4] A. Lumsdaine, D. Gregor, B. Hendrickson, and J. Berry, "Challenges in parallel graph processing," Parallel Processing Letters, 2007.

[5] A. Janin, "Speech recognition on vector architectures," Ph.D. dissertation, Univer-sityof California, Berkeley, Berkeley, CA, 2004.

[6] H. Ney and S. Ortmanns, "Dynamic programming search for continuous speech recognition," IEEE Signal Processing Magazine, vol. 16, pp. 64–83, 1999.

[7] M. Ravishankar, "Parallel implementation of fast beam search for speaker-independent continuous speech recognition," 1993.

[8] S. Phillips and A. Rogers, "Parallel speech recognition," Intl. Journal of Parallel Programming, vol. 27, no. 4, pp. 257–288, 1999.

[9] K. You, Y. Lee, and W. Sung, "OpenMP-based parallel implementation of acontinousspeech recognizer on a multi-core system," in Proc. IEEE Intl. Conf.on Acoustics, Speech, and Signal Processing (ICASSP), Taipei, Taiwan, 2009.

[10] M. Mohri, F. Pereira, and M. Riley, "Weighted finite state transducers in speech recognition," Computer Speech and Language, vol. 16, pp. 69–88, 2002.

[11] S. Ishikawa, K. Yamabana, R. Isotani, and A. Okumura, "Parallel LVCSR algo-rithmfor cellphone-oriented multicore processors," in Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), Toulouse, France, 2006.

[12] P. R. Dixon, T. Oonishi, and S. Furui, "Fast acoustic computations using graphics processors," in Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (ICASSP), Taipei, Taiwan, 2009.

[13] P. Cardinal, P. Dumouchel, G. Boulianne, and M. Comeau, "GPU accelerated acoustic likelihood computations," in Proc. Interspeech, 2008.

[14] J. Chong, Y. Yi, N. R. S. A. Faria, and K. Keutzer, "Data-parallel large vocabulary continuous speech recognition on graphics processors," in Proc. Intl. Workshop on Emerging Applications and Manycore Architectures, 2008.

[15] S. Kumar, C. J. Hughes, and A. Nguyen, "Carbon: Architectural support for fine- grained parallelism on chip multiprocessors," in Proc. Intl. Symposium on Computer Architecture (ISCA), 2007.

[16] NVIDIA CUDA Programming Guide, NVIDIA Corporation, 2009, version 2.2 beta. [Online]. Available: http://www.nvidia.com/CUDA

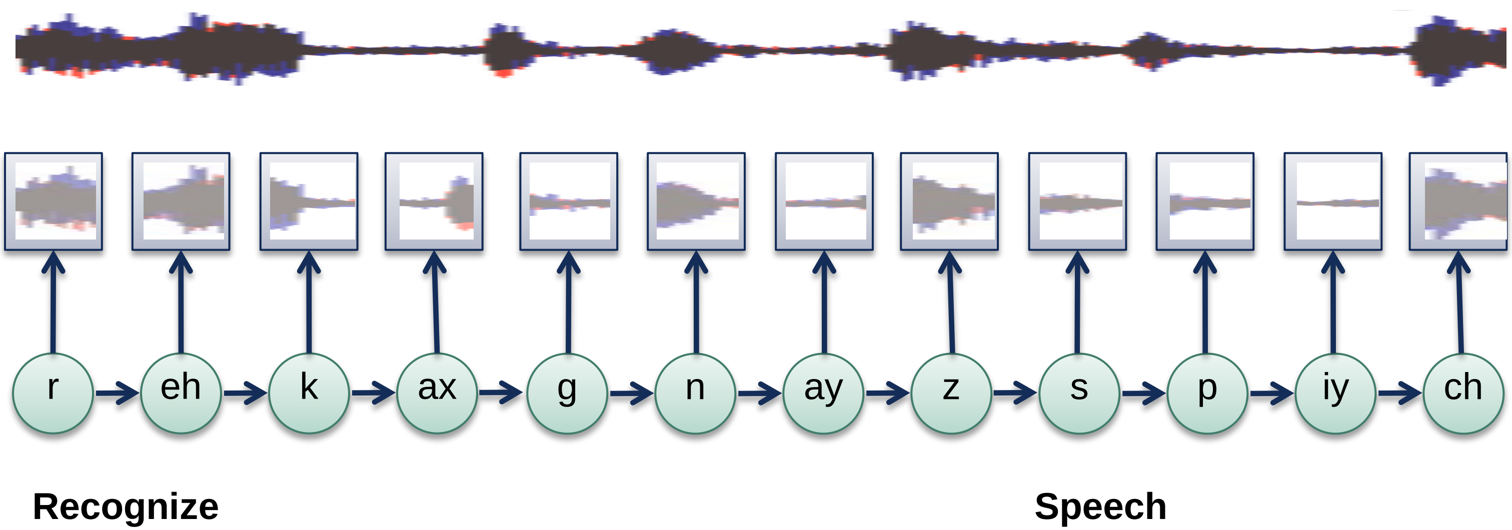
[17] G. T. et al, "The CALO meeting speech recognition and understanding system," in Proc. IEEE Spoken Language Technology Workshop, 2008, pp. 69–72.

[18] A. Stolcke, X. Anguera, K. Boakye, O. Cetin, A. Janin, M. Magimai-Doss, C. Wooters, and J. Zheng, "The SRI-ICSI spring 2007 meeting and lecture recognition system," Lecture Notes in Computer Science, vol. 4625, no. 2, pp. 450–463, 2008.

Backup Slides

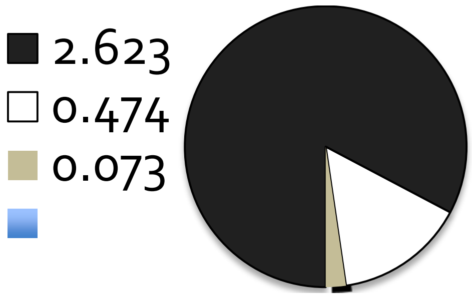
Hidden Markov Model

- In the Hidden Markov Model, states are ***hidden***, because phones are ***indirectly observed***
- One must infer the ***most likely interpretation*** of the signal while taking the model of the ***underlying language*** into account

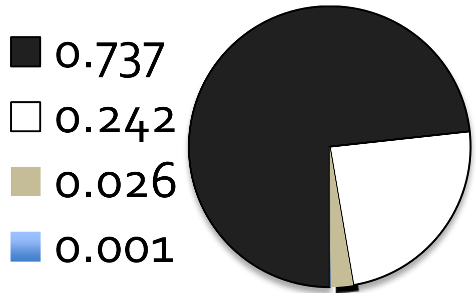


Detailed Speedup: Multicore

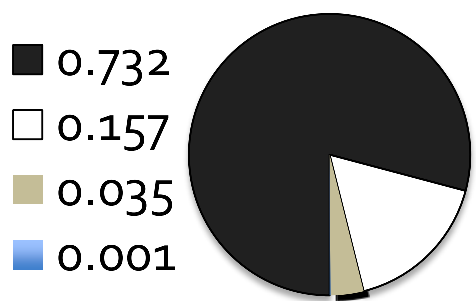
Sequential
RTF: 3.17; 1x



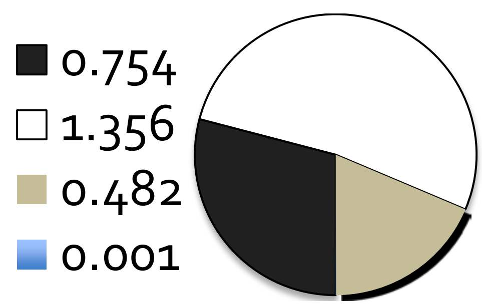
Arc-based Propagation
RTF: 1.006; 3.2x



State-based Propagation
RTF: 0.925; 3.4x



State-based Aggregation
RTF: 2.593; 1.2x



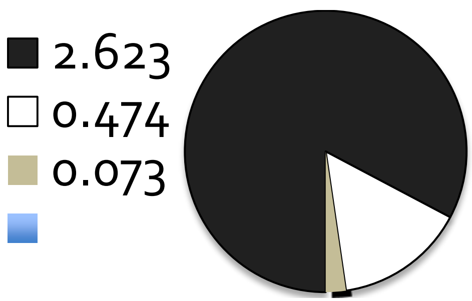
RTF: Real Time Factor

3.4x: vsSeq Speedup

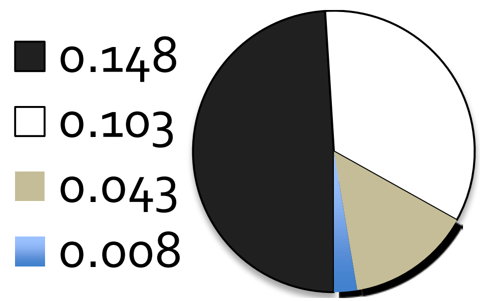
Phase 1
Phase 2
Phase 3
Seq. Overhead

Detailed Speedup: Manycore

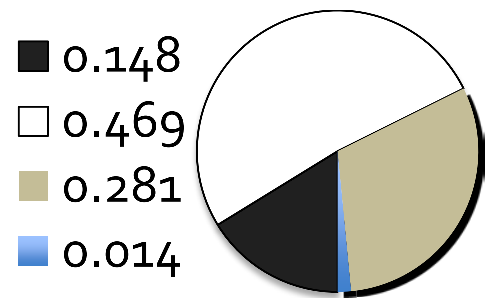
Sequential
RTF: 3.17; 1x



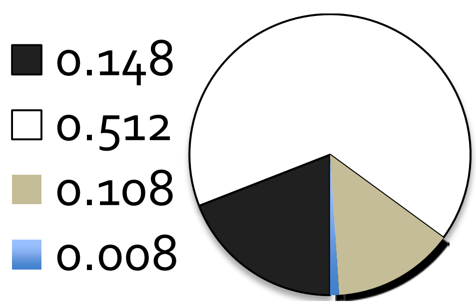
Arc-based Propagation
RTF: 0.302; 10.5x



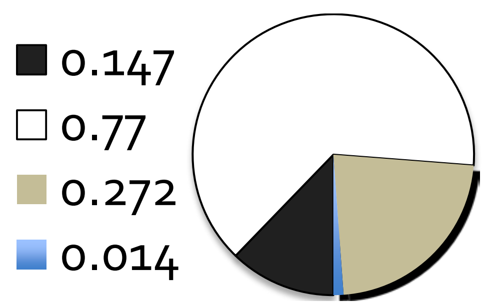
Arc-based Aggregation
RTF: 0.912; 3.5x



State-based Propagation
RTF:0.776; 4.1x



State-based Aggregation
RTF: 1.203; 2.6x



RTF: Real Time Factor

3.4x: Speedup vsSeq

Phase 1
Phase 2
Phase 3
Seq. Overhead

Next Steps

- Experiment on two more sets of models
  - Telephone conversations (optimizing for batch model processing)
  - News Broadcast (optimizing for real time processing)

- Construct the application framework for domain experts to develop speech applications
  - Search for industry use cases to substantiate usage scenarios

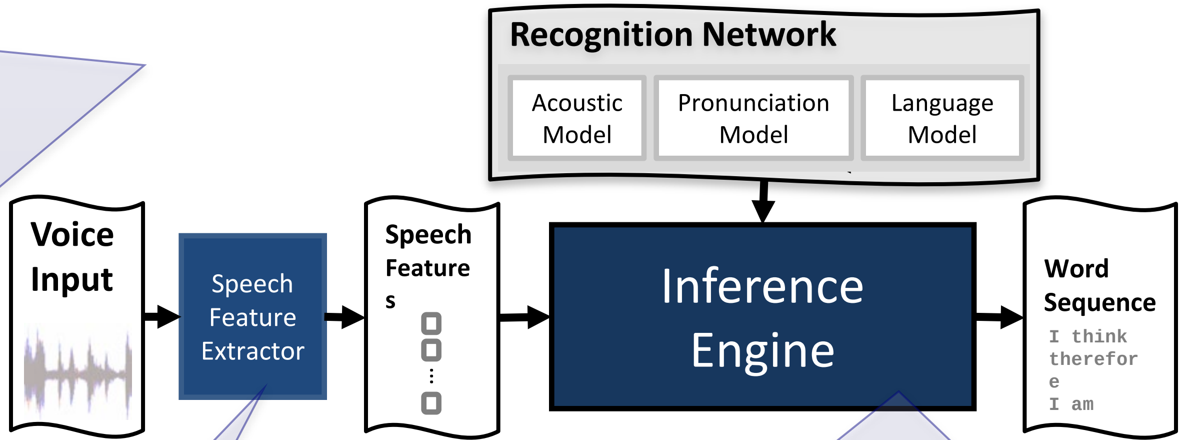
LVCSR Application Framework

Top Level Attributes

Customizable attributes:
- Recognition network structure
- Input waveform format
- Output word sequence format

Data Structure:
- Feature vector format

Fixed Structure:
- Feature extractor
- Inference engine



Feature Extractor

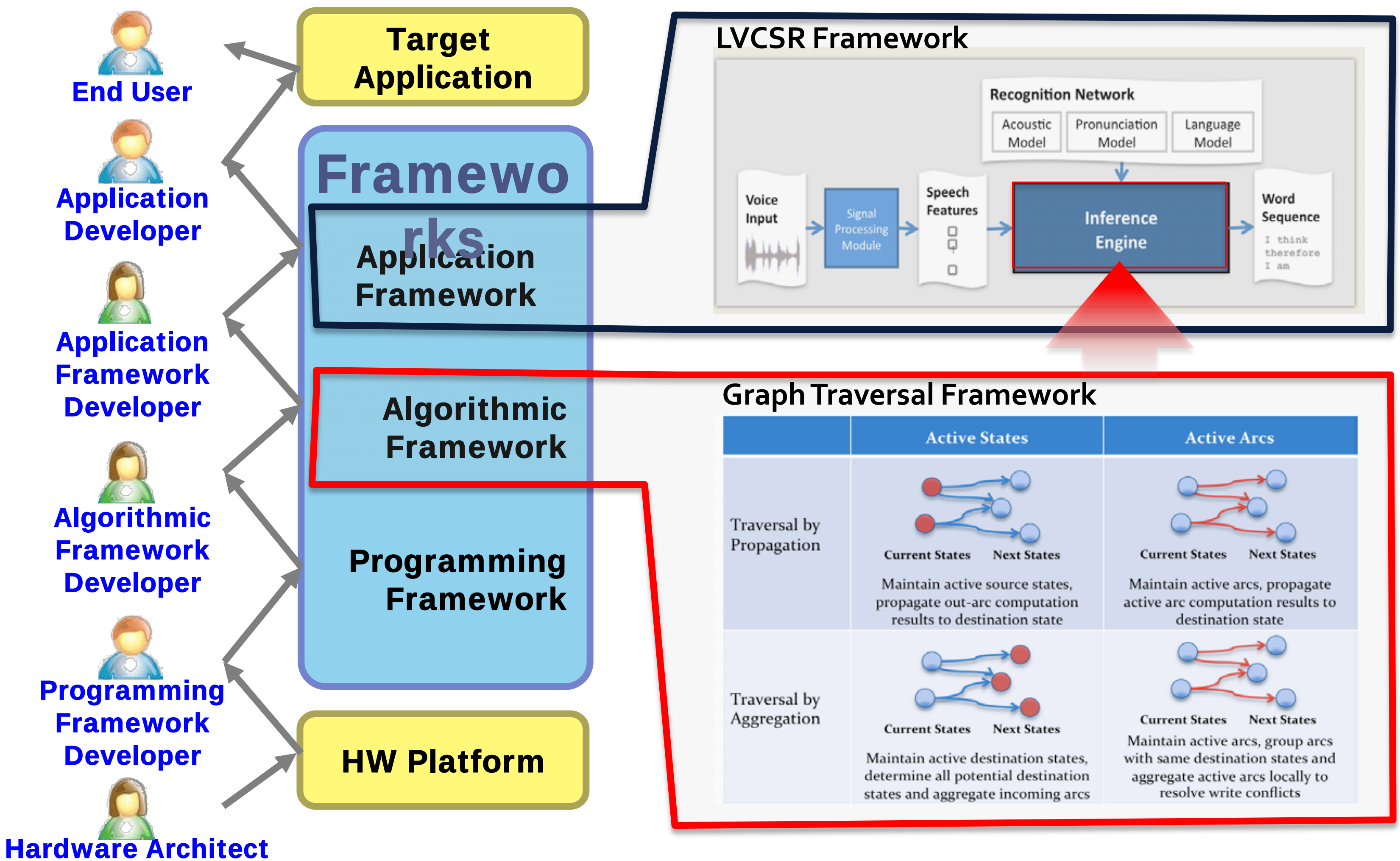
Customizable Function:
- Feature extraction algorithm

Inference Engine

Customizable Functions:
- Observation/Arc probability computation
- Pruning heuristics
- Track back data logging

Framework architecture customization:
- States vs arc based traversal
- Propagate vs aggregate traversal techniques

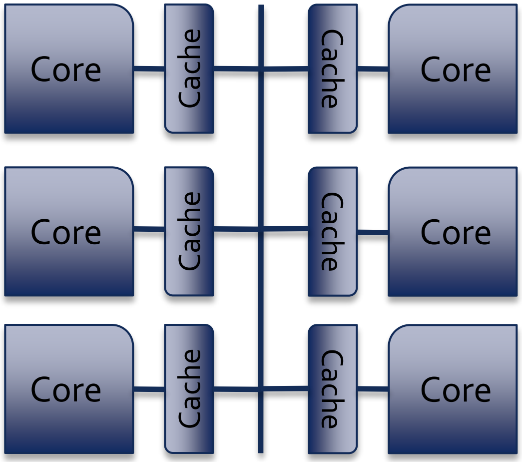
Frameworks for Parallel Programming



Discussion: Load Balancing

- Core level load balancing is an important issue
  - Many prior work has been limited by across core work load imbalance
- Application developers want to expose parallelism, not managing the detail
  - Best solved by implementation platform support

- Multicore:
  - Task queue abstraction with distributed queue and lazy work stealing [15]
- Manycore:
  - Hardware managed dynamic load balancing based on the CUDA runtime environment [16]



[15] S. Kumar, C. J. Hughes, and A. Nguyen, "Carbon: Architectural support for fine-grained parallelism on chip multiprocessors," in Proc. Intl. Symposium on Computer Architecture (ISCA), 2007.
[16] NVIDIA CUDA Programming Guide, NVIDIA Corporation, 2009, version 2.2 beta. [Online]. Available: http://www.nvidia.com/CUDA

Discussion: Memory Hierarchy

Intel Core i7

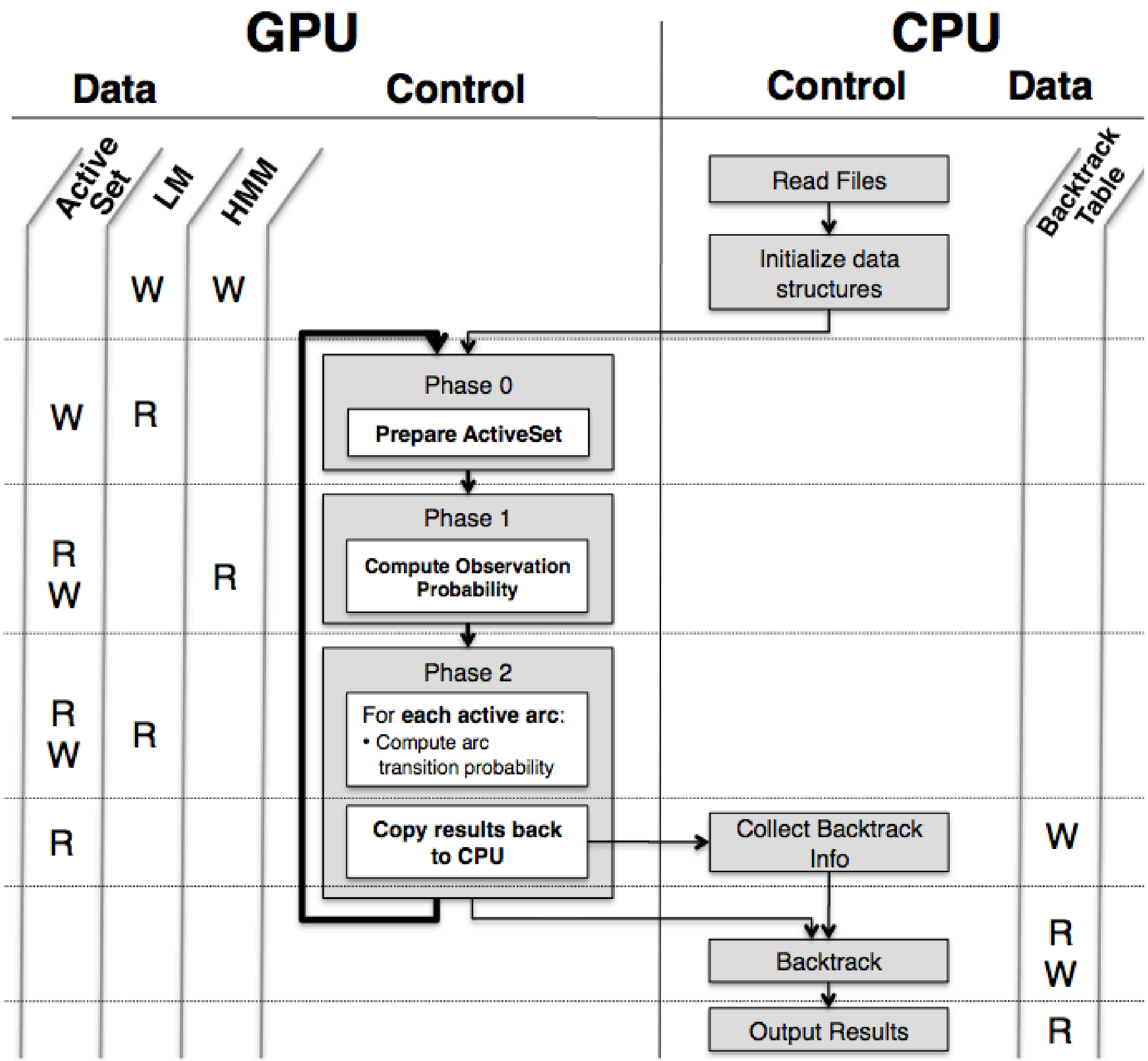
	Bandwidth	Size
L1	*340 GB/s	32KB Data 32KB Inst
L2	*170 GB/s	256KB per core
L3	-	8MB
DRAM	25.6 GB/s	6GB (24GB max)

NVIDIA GTX 280

	Bandwidth	Size
Shared Memory	1244 GB/s	16KB Data per SM unit
GDDR	141.7 GB/s	1 GB
PCI Express	2.5 GB/s	Up to 24 GB

- Currently, the memory hierarchy differs significantly between Intel multicore and NVIDIA manycore
  - Requires different data structure for optimal performance
- Multicore:
  - Reference data in main memory, working set mostly cached in L3
- Manycore:
  - Create temporary coalesced array for working set, stored in GDDR, streaming access

Speech Inference Engine Implementation

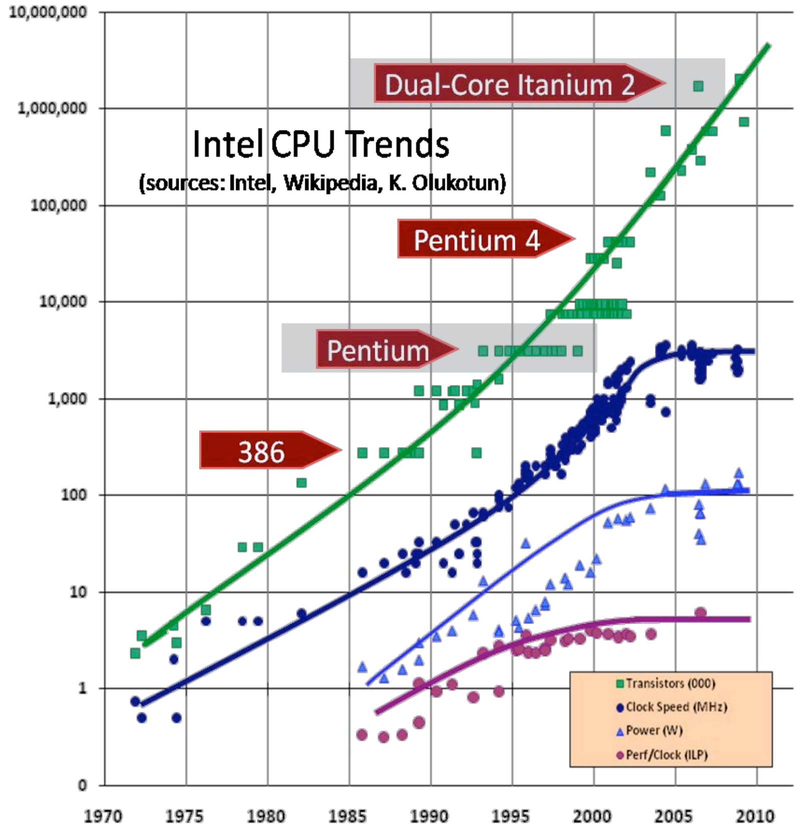


Recognition Network Representation

- Linear-Lexical Model (LLM) – baseline implementation
  - Models each word as a chain of triphone states
  - Highly redundant
  - Language model from word-to-word transitions
- Weighted Finite State Transducer (WFST)
  - Combines pronunciation and language models
  - Takes advantage of sparsity of natural languages
  - Remove redundant states and arcs
  - Faster recognition speed on *sequential* processors

Software Must Use Hardware Parallelism

Hardware Trends



Software Trends