Optimizing a Semantic Comparator using CUDA-enabled Graphics Hardware

Aalap Tripathy

Suneil Mohan, Rabi Mahapatra

Embedded Systems and Codesign Lab

codesign.cse.tamu.edu

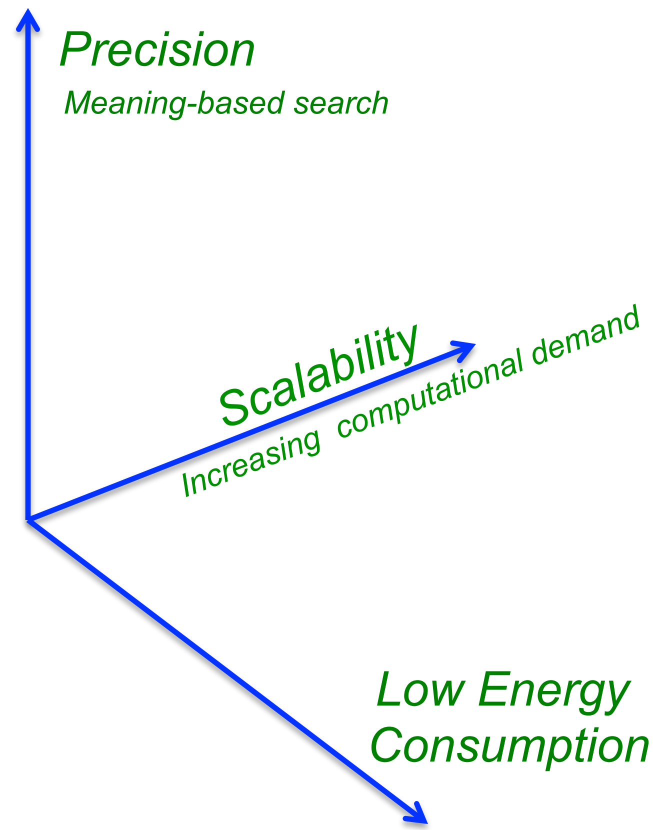
(Presented at ICSC 2011, September 19, 2011 in Palo Alto, CA)

Overview

- Introduction
- Current v/s future technologies
- Key steps in computation
- Description of architecture
- Experimental Setup & Results
- Conclusion

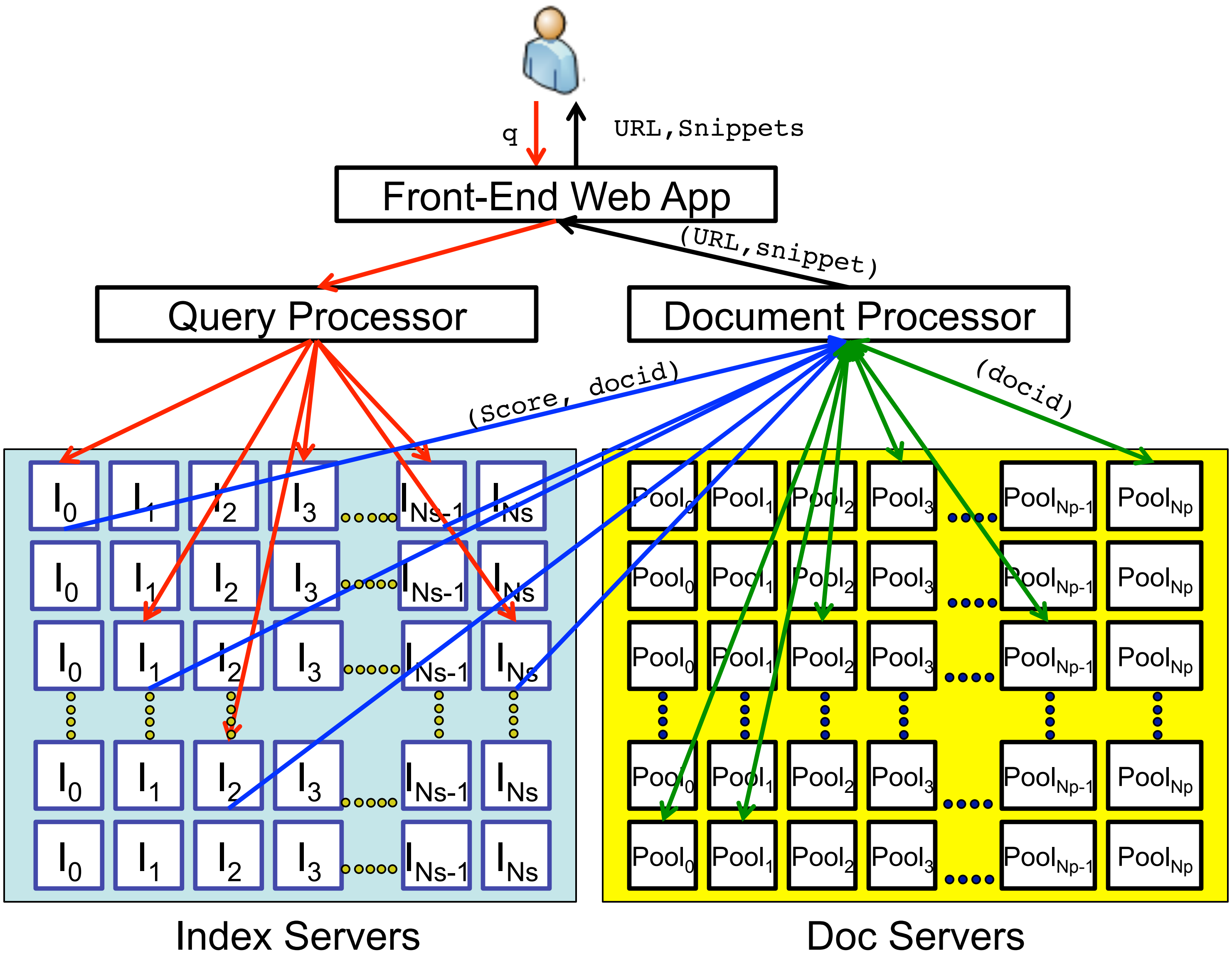
Introduction

- Search is a key activity on the Internet
  - 13 billion queries a month (3500/sec)
  - Growing rapidly (38% annually)
- Search Engines need to be Precise
  - Increased user expectations from search results
  - Not 200 links but few relevant documents
- Search Engines need to be Scalable
  - Search engines deployed as distributed systems
  - Newer methods make more computational demand
- Search Engines need to consume Low Energy
  - Tens of Mega Watts (12.5 MW/year)
  - Coarse-grained, task-parallel approach is insufficient



- Objective: Deploy meaning-based search to enhance search-quality while consuming less energy and meeting time constrains

Current Search Engines



Current Search Paradigm – Vector Method

"The American man ate Indian food."

$$D^{1} = s^{1}_{\mathrm{American}}\vec{V}_{\mathrm{American}} + s^{1}_{\mathrm{Man}}\vec{V}_{Man} + s^{1}_{ate}\vec{V}_{ate} + s^{1}_{Indian}\vec{V}_{Indian} + s^{1}_{Food}\vec{V}_{Food}$$

Descriptor representing meaning of the entire text

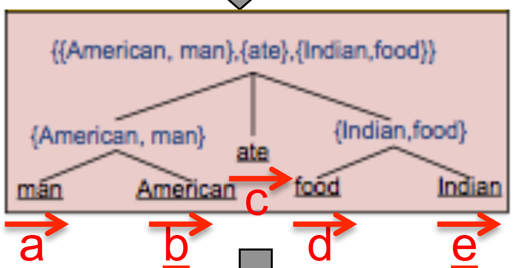
Basis vector representing the term "American"

Scalar weight / coefficient denoting relative importance of the term

- Vector Methods can not differentiate between two documents containing the same keywords
  - "American man ate Indian food" v/s "Indian man ate American food"
  - Produces hundreds of irrelevant results – "no precision"
    - Hundreds of redundant operations performed in the process
    - Tens of MW of power consumed in the process

Future Search paradigm - Tensor Method

The American man ate Indian food



$$s_1 \overrightarrow{\triangleright\triangleright a b \triangleleft c \triangleleft} + s_2 \overrightarrow{\triangleright a c \triangleleft} + s_3 \overrightarrow{\triangleright b c \triangleleft} +$$

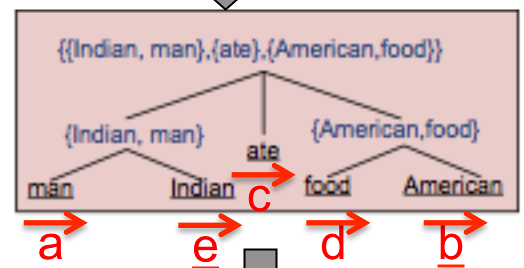
$$s_4 \overrightarrow{\triangleright a b \triangleleft} + s_5 \vec{a} + s_6 \vec{b} + s_7 \vec{c} + \cdots \text{(31 Terms)}$$

Basis vector terms

$\vec{a} = \text{"man"}$, $\vec{b} = \text{"american"}$, $\vec{c} = \text{"ate"}$, $\overrightarrow{\triangleright a b \triangleleft} = \text{"}\triangleright \text{ man american } \triangleleft\text{"}$,

$\overrightarrow{\triangleright b c \triangleleft} = \text{"}\triangleright \text{ american ate } \triangleleft\text{"}$, $\overrightarrow{\triangleright\triangleright a b \triangleleft c \triangleleft} = \text{"}\triangleright\triangleright \text{ man american } \triangleleft \text{ ate } \triangleleft\text{"}$,

The Indian man ate American food



$$s_1 \overrightarrow{\triangleright\triangleright a e \triangleleft c \triangleleft} + s_2 \overrightarrow{\triangleright a c \triangleleft} + s_3 \overrightarrow{\triangleright e c \triangleleft} +$$

$$s_4 \overrightarrow{\triangleright a e \triangleleft} + s_5 \vec{a} + s_6 \vec{e} + s_7 \vec{c} + \cdots \text{(31 Terms)}$$

Basis vector terms

$\vec{a} = \text{"man"}$, $\vec{e} = \text{"Indian"}$, $\vec{c} = \text{"ate"}$, $\overrightarrow{\triangleright a e \triangleleft} = \text{"}\triangleright \text{ man Indian } \triangleleft\text{"}$,

$\overrightarrow{\triangleright e c \triangleleft} = \text{"}\triangleright \textit{ Indian} \text{ ate } \triangleleft\text{"}$, $\overrightarrow{\triangleright\triangleright a e \triangleleft c \triangleleft} = \text{"}\triangleright\triangleright \text{ man Indian } \triangleleft \text{ ate } \triangleleft\text{"}$,

- Tensor methods differentiate documents containing same keywords
  - Captures the relationship between terms
  - At what cost? – Exponentially larger number of terms

Semantic Comparison using Tensors

(The american man ate indian food)

$$s^1_1 \overrightarrow{\triangleright\triangleright ab\triangleleft c\triangleleft} + s^1_2 \overrightarrow{\triangleright ac\triangleleft} + s^1_3 \overrightarrow{\triangleright bc\triangleleft} + s^1_4 \overrightarrow{\triangleright ab\triangleleft} + s^1_5 \vec{a} + s^1_6 \vec{b} + s^1_7 \vec{c} + \cdots$$

Tensor (T1)

(The indian man ate american food)

$$s^2_1 \overrightarrow{\triangleright\triangleright ae\triangleleft c\triangleleft} + s^2_2 \overrightarrow{\triangleright ac\triangleleft} + s^2_3 \overrightarrow{\triangleright ec\triangleleft} + s^2_4 \overrightarrow{\triangleright ae\triangleleft} + s^2_5 \vec{b} + s^2_6 \vec{c} + s^2_7 \vec{a} + \cdots$$

Tensor (T2)

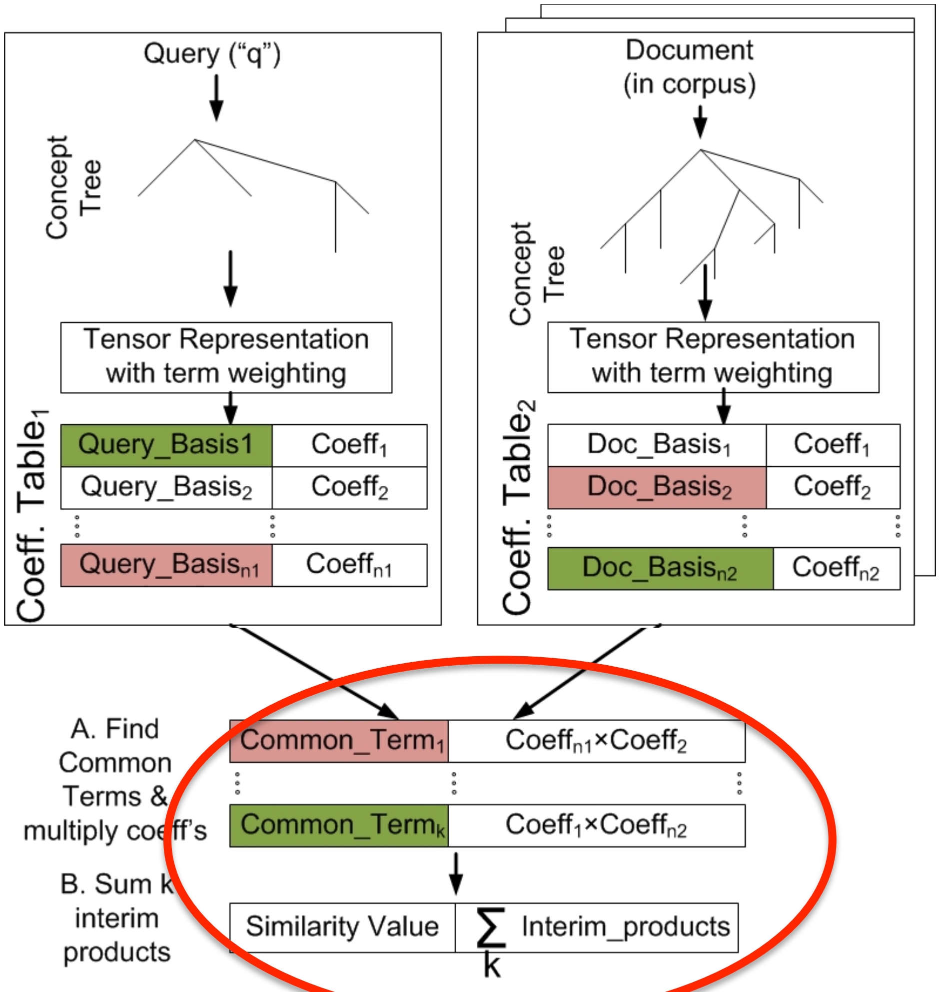
$$\mathrm{Similarity}(T_1, T_2) = T_1 \bullet T_2 = s^1_5 s^2_7 + s^1_6 s^2_5 + s^1_7 s^2_6 \qquad (<1)$$

1. Identify common basis vectors
2. Multiply scalar coefficients
3. Find sum of all products

Overview

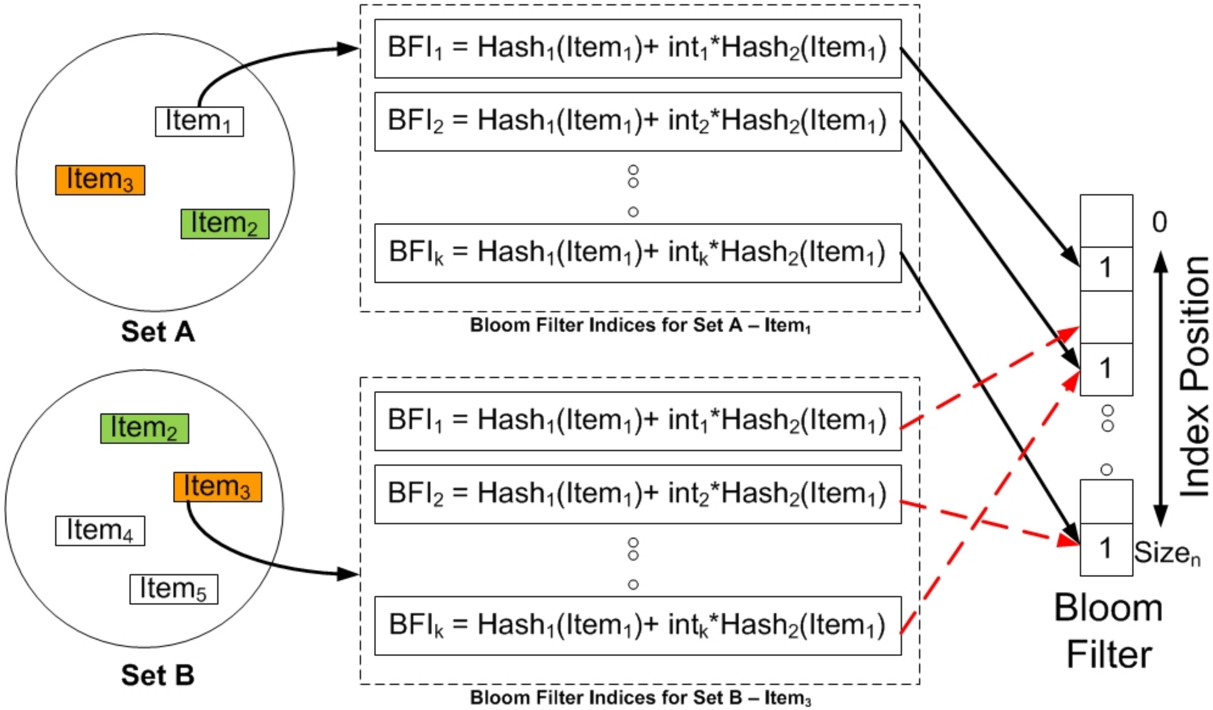
- Introduction
- Current v/s future technologies
- Key steps in computation
- Challenges
- Description of architecture
- Experimental Setup & Results
- Conclusion

Key Steps in Semantic Computation



- For two Tensors of size n_1, n_2, Search is $O(n_1.n_2)$ or $O(n_1.\ log\ n_2)$
- Can we improve upon this?

Bloom Filters



- Bloom Filter – Enables Compact representation of a Set
  - Parameters:
    - Number of elements to be inserted (m)
    - Size of the Bloom Filter ($size_n$)
    - Number of Indices used to represent each element (k)
  - Probability of false positives can be controlled

Details of Comparison

Coefficient table of Query Tensor (Table 2)

Tensor id	Coeffs	Set of BF bit indices
Id_1	s_1	$\{ x_i : 0 \le x_i \le size_n \}$
Id_i	**$s_i = 0.2$**	**{ 0, 2,…5 }**
Id_n	s_n	{…}

Coefficient table of Doc Tensor (Table 1)

Tensor id	Coeffs	Set of BF bit indices
Id_1	s_1	$\{ x_i : 0 \le x_i \le size_n \}$
Id_i	**$s_i = 0.4$**	**{ 0, 2,…5 }**
Id_n	s_n	{…}

Bloom filter of Doc Tensor

Bit	Index
1	0
0	1
1	2
0	3
0	4
⋮	
1	$Size_n$-2
0	$Size_n$-1

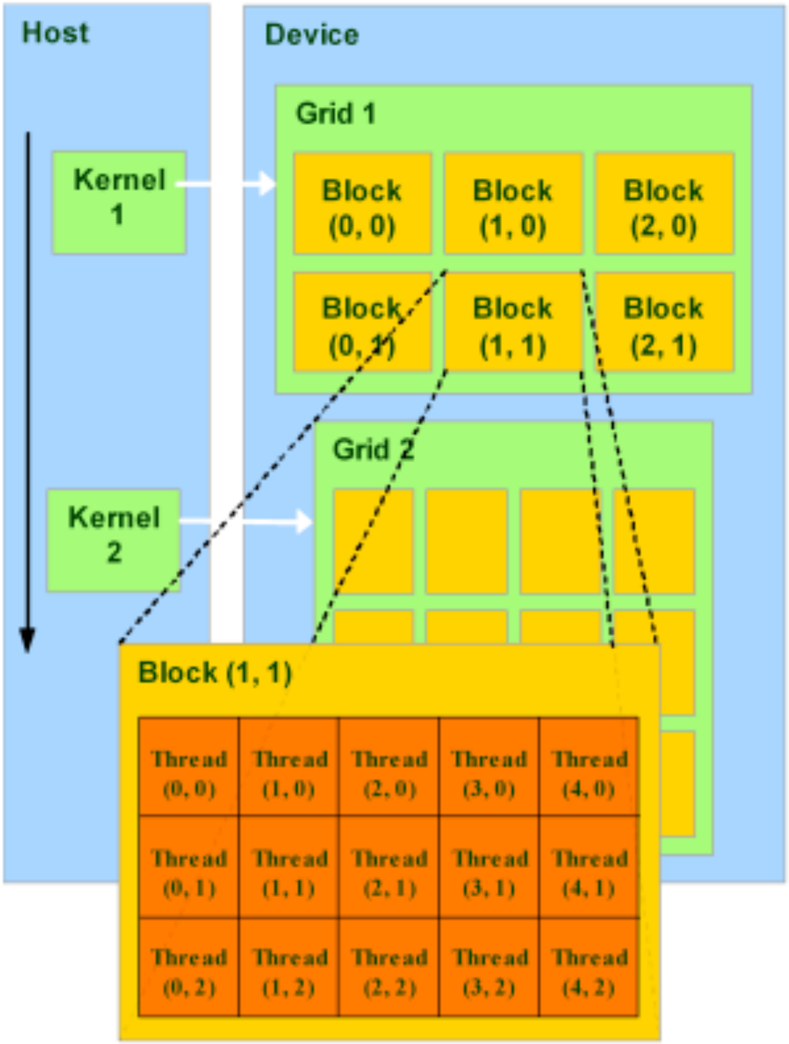
$$T_1 \bullet T_2 = \ldots + s_i^1 s_i^2 + \ldots + s_j^1 s_j^2 + \ldots$$

1. Identify common basis vectors (filtered)
2. Multiply coefficients
3. Compute sum of products

Overview

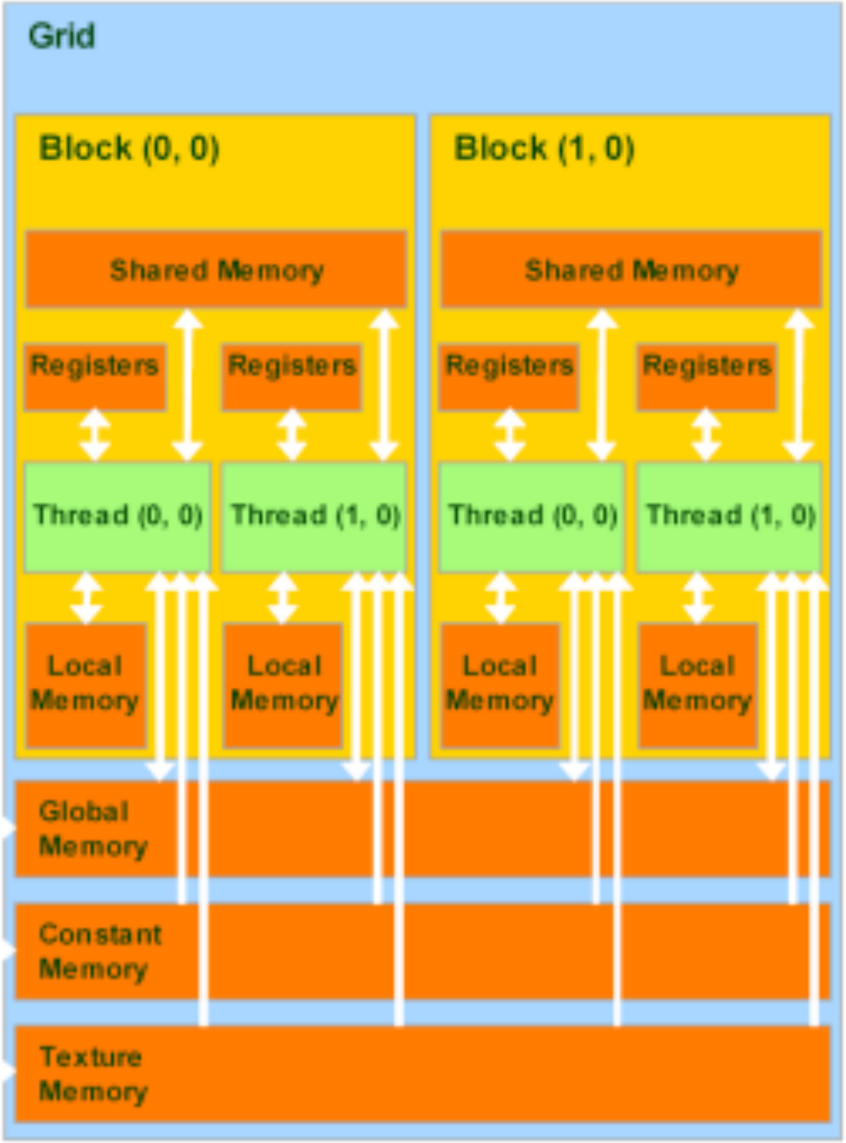
- Introduction
- Current v/s future technologies
- Key steps in computation
- Challenges
- Description of architecture
- Experimental Setup & Results
- Conclusion

Architecture Description - CUDA



- CUDA
  - Compute Unified Device Architecture
    - Device Architecture spec
    - An extension to C (library, API, compiler)
- GPGPU uses heterogeneous parallel computing model
  - Kernel is called by host and run by GPU
  - Each SIMD processor executes same instruction over different data elements in parallel
  - Can process thousands of threads simultaneously.
  - The number of logical threads and thread blocks surpasses the number of physical execution units

Architecture Description – CUDA Memory Model



- CUDA Memory Model
  - Threads
    - Registers (per thread)
    - Local Memory (off-chip)
  - Blocks
    - Shared Memory between threads
  - Device
    - Global Memory
      - between kernels
    - Constant Memory
      - Read only, store invariants
    - Texture Memory
      - limited but can cache parts of Global Memory

Semantic Comparison using CUDA

TEXAS A&M UNIVERSITY

Phase A
- Copy Table1 from Host PC to CUDA Global Memory
- Copy Table 2 from Host PC to CUDA Global Memory

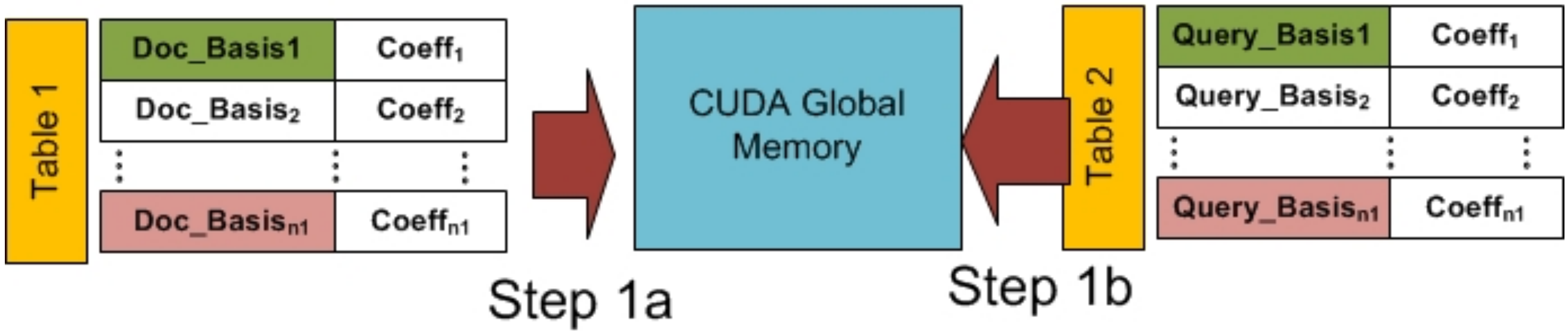
Phase B
- Encode Table 1 in Bloom Filter

Phase C
- Encode Table 2. Test presence in Bloom Filter

Phase D
- Compute Semantic Similarity using Filtered elements

- CUDA Programming Model
  - Split a task into subtasks
  - Divide input data into chunks that fit global memory
  - Load a data chunk from global memory into shared memory
  - Each data chunk is processed by a thread block
  - Copy results from shared memory back to global memory
- **Optimization 1:** Maximize independent parallelism

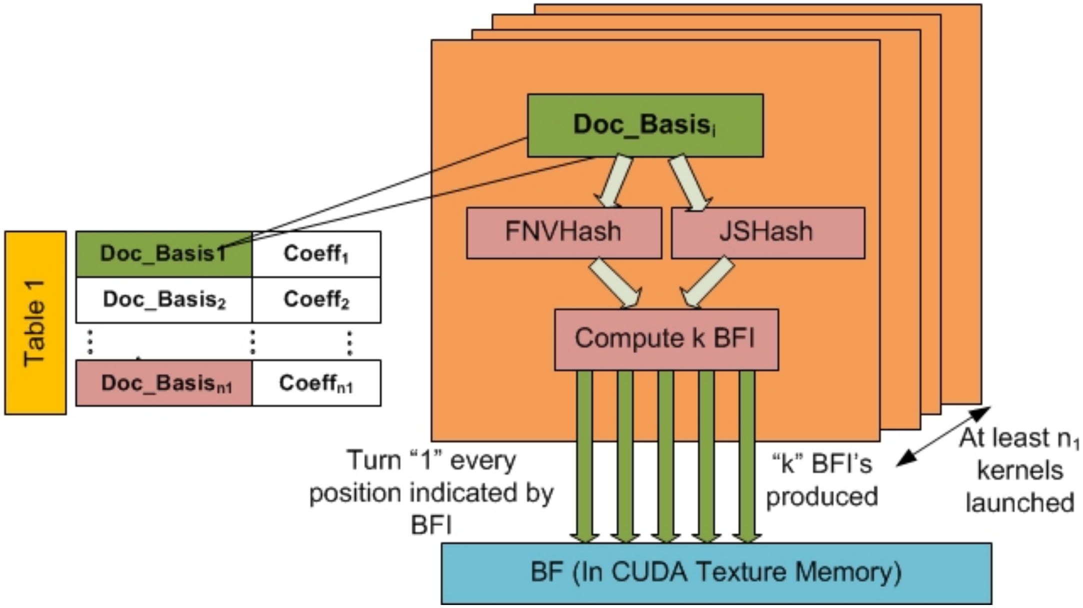
Phase A – Copy Data from Host to GPU



- Copy two tables to be compared into CUDA global Memory
  - Data has to be explicitly copied into CUDA Global Memory
  - **Optimization 2:** Data structure is flattened to increase coalesced memory accesses
  - Maximize the available PCIe bandwidth (76.8 Gb/s for NVIDIA C870)

Phase B – Encode Co-efficient Table 1 in Bloom Filter

TEXAS A&M UNIVERSITY

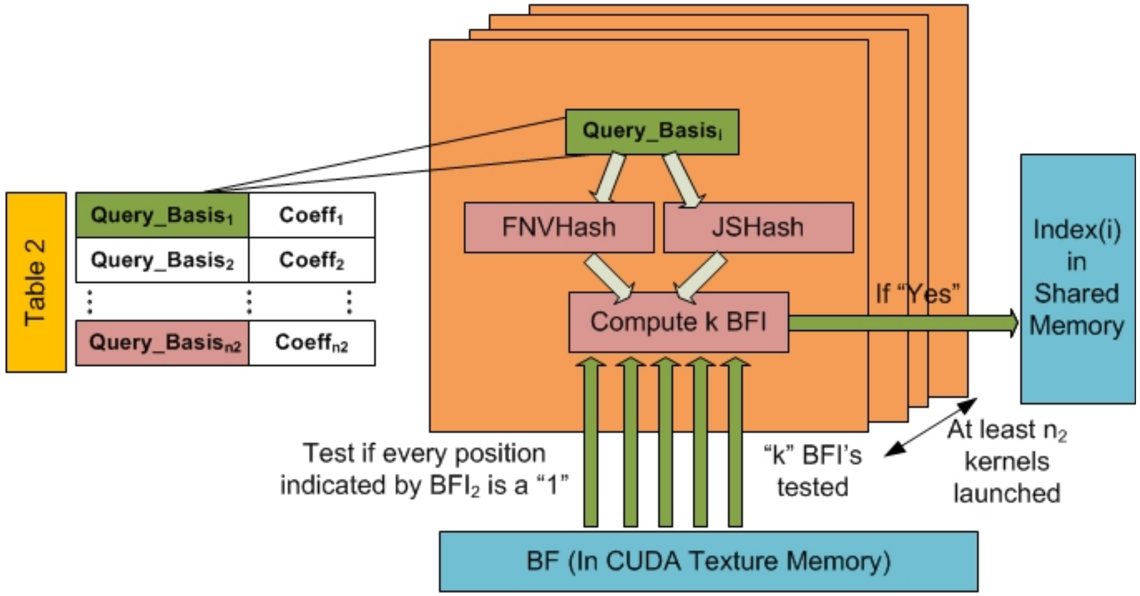


- Encode Table1 (Document Basis Coefficient Table) in Bloom Filter
  - The i^{th} Doc_Basis term is hashed using two hash functions
  - "k" additional Bloom Filter Indices are generated using:

$$BFI_k = Hash_1(Item) + int_k \times Hash_2(Item)$$

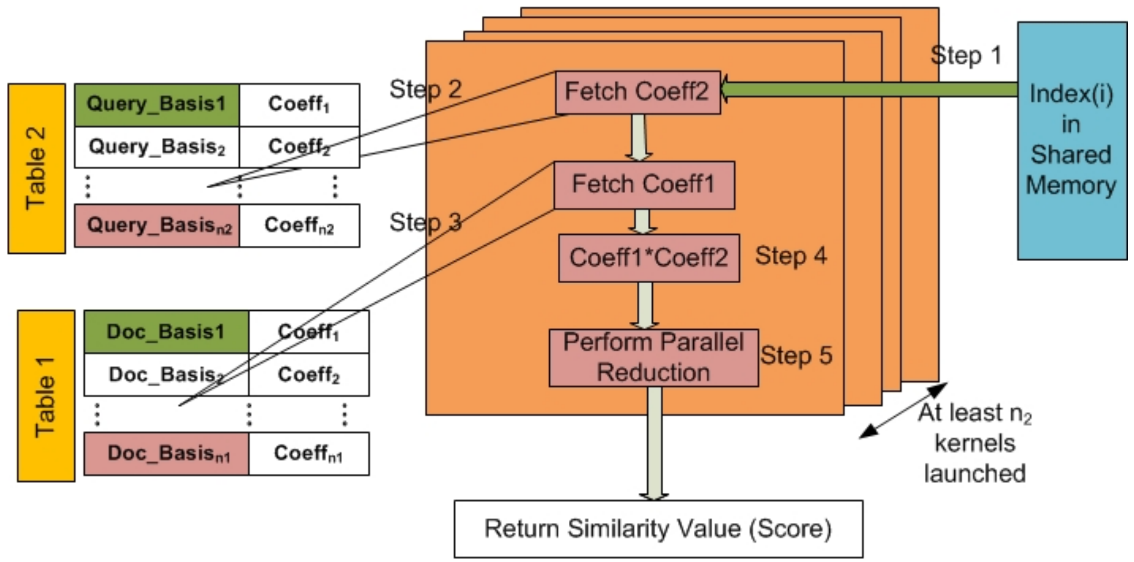
  - Turn every Index Position "1" in BF bit array in CUDA texture Memory
  - **Optimization 3:** At least n_1 threads are launched. Limit the number of blocks and increase the number of threads. Increases shared memory reuse.

Phase C – Encode & Test Table 2 using Bloom Filter



- Encode Table2 in Bloom Filter, Test
  - The i^th Query_Basis term is hashed using same two hash functions
  - "k" additional Bloom Filter Indices are generated
  - Those index positions are tested in BF bit array in CUDA texture Memory
  - At least n_2 threads are launched.
- If all indices are "1", Query_Basis_i is a "**filtered element**", store Index (i)
- **Optimization 4:** Shared memory is inaccessible after end of kernel. This data is transferred to Global Memory at the end of each thread block

Phase D – Compute Semantic Similarity using Filtered elements



- Extract corresponding scalar coefficients, multiply and sum
  - The index of the potential match in Table 2 is used to lookup **Coeff$_2$**
  - The corresponding match in Table 1 is used to lookup **Coeff$_1$**
  - The same kernel performs multiplication (interim products)
  - Intermediate products from multiple threads are summed in parallel.

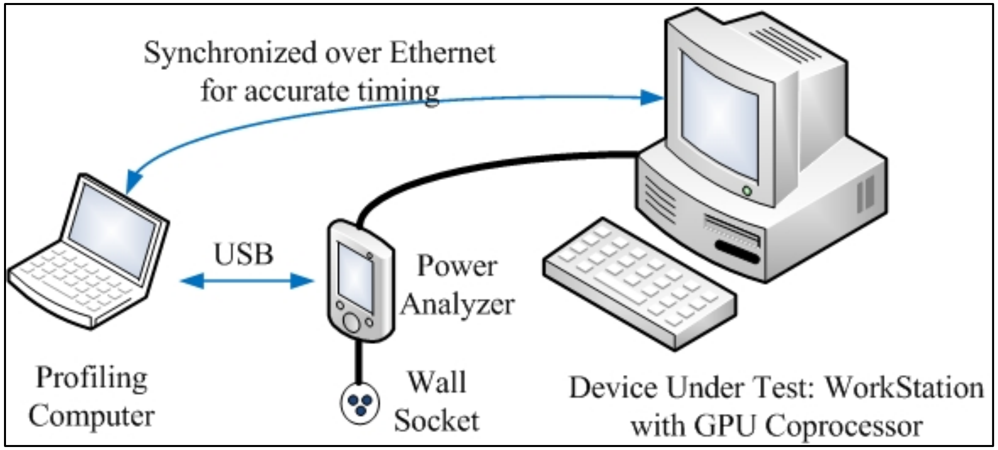
Other Algorithmic Optimizations

- Partitioning the computation to keep all stream cores busy
  - **Optimization 5:** Multiple threads, multiple thread blocks in constant use
- Monitoring per-processor resource utilization
  - **Optimization 6:** Low utilization per thread block allows multiple active blocks per multi-processor

Overview

- Introduction
- Current v/s future technologies
- Key steps in computation
- Description of architecture
- Experimental Setup & Results
- Conclusion

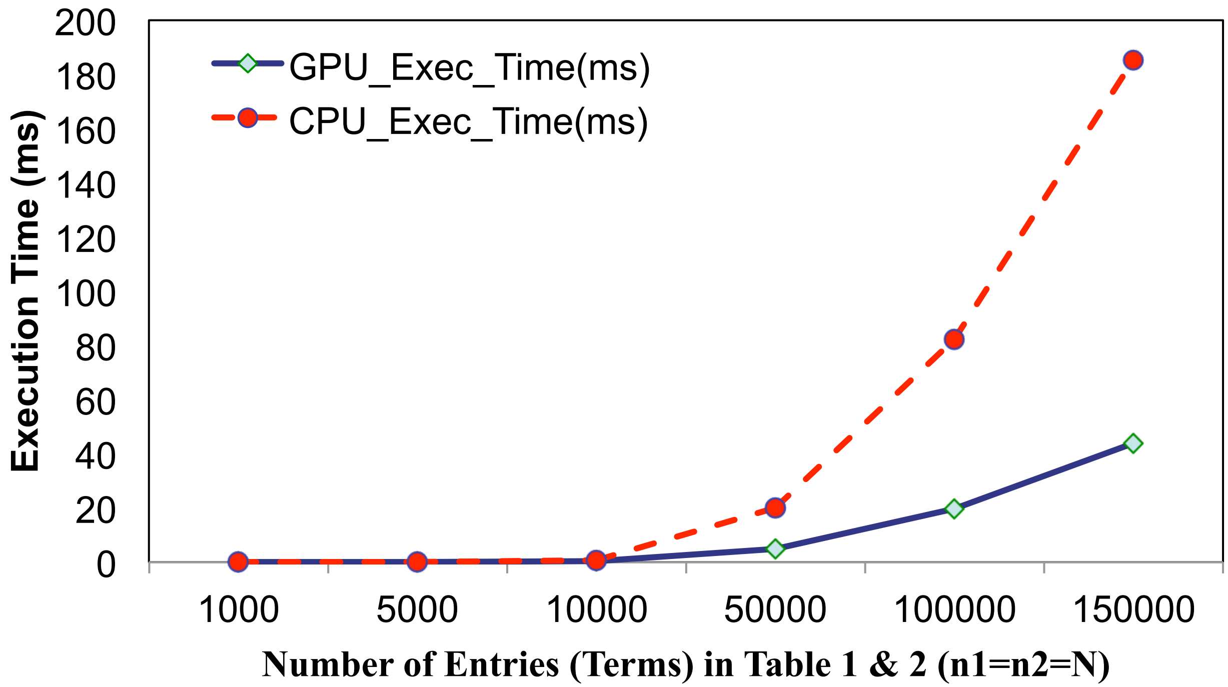
Experimental Setup



- Experimental setup
- Experimental Parameters
  - Table Sizes (N)
  - Similarity between Tables (c)
  - CUDA Parameters (num_bocks, threads / block)
- Experimental Measurements
  - Execution Time
  - Power/Energy
  - Throughput (Comparisons / sec)

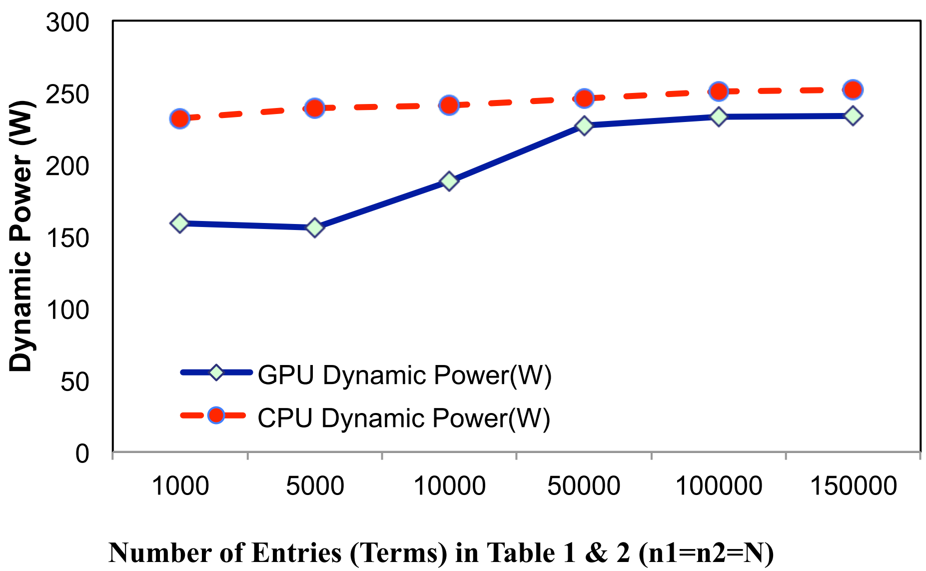
Device Characteristics	Values
GPU – # Stream Processors / cores	**128/16 (Nvidia Tesla C870)**
Core Frequency	600 Mhz
CUDA Toolkit	3.1
Interface	16x PCI-Express
Memory Clock	1.6 GHz
Global Memory	**1.6GB**
Constant Memory	65KB
Shared Memory/block	16KB
Registers per block	8192
Number of threads per block	512
Memory Bus Bandwidth	76.8 GB/s, 384 bit-wide GDDR3
Warp Size (Number of threads per thread processor)	32
CPU – P4	2 GB RAM, Ubuntu 9.10

Results - Execution Time Profiling



- Exponential increase in CPU execution time for large tables
- Same dataset on a GPU is up to 4x faster (similarity c=10%)

Results - Power Profiling



CPU-GPU Power Characteristic	Value
System Base Power	115W
System Idle Power (GPU cold shutdown)	150W
System Idle Power (GPU Awake, Idle)	186W
GPU Idle Power	36W

- Measured using WattsUp Power Meter. (Measures Mains Power)
- GPU dynamic power is lower but approaches that of a CPU for N>50000
- GPU's are known to be energy-efficient but not necessarily power-efficient

Results - Energy Saved per Comparison

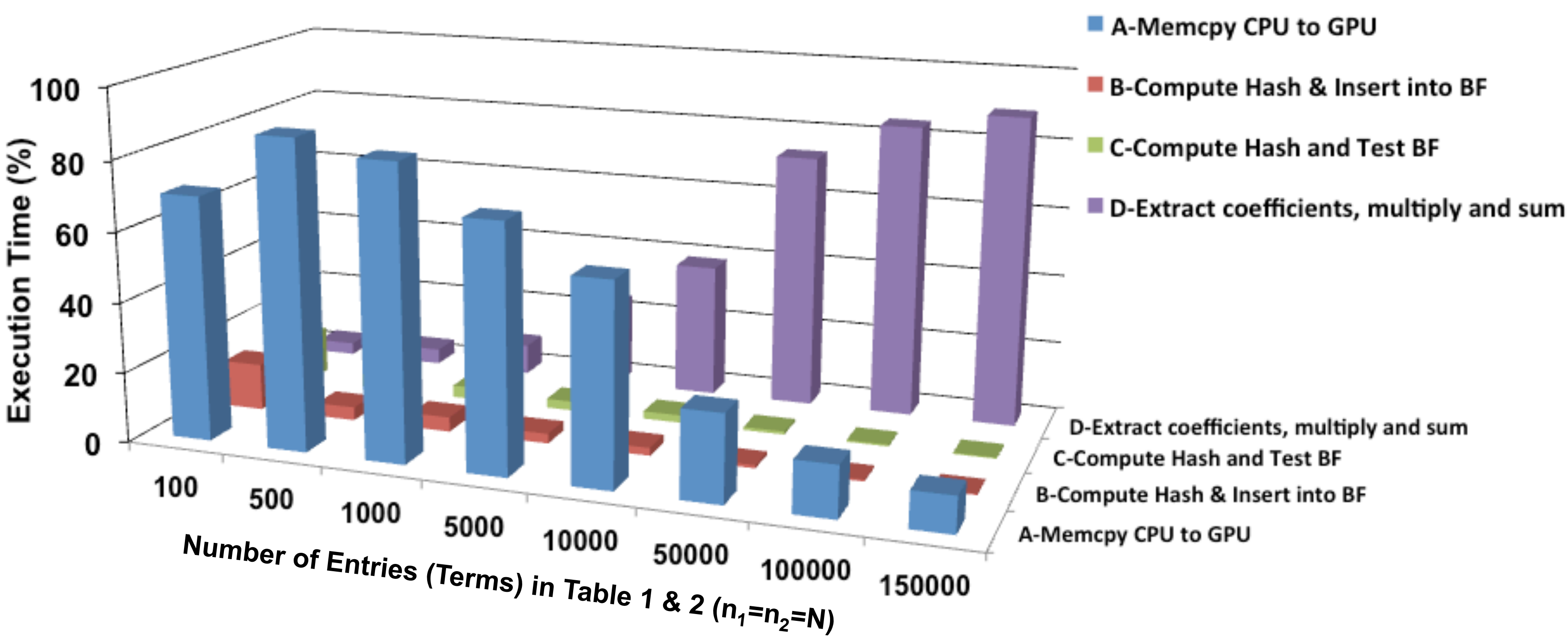
TEXAS A&M UNIVERSITY

Table Size (N)	CPU Execution Time (s)	CPU Average Power (W)	GPU Execution Time (s)	GPU Average Power (W)	Energy Saved (%)
5k	0.18	232	0.05	159	**79.65**
10k	0.74	239	0.21	156	**77.64**
50k	20.0	241	4.93	188	**77.27**
100k	82.4	246	19.57	227	**77.96**
150k	185.3	251	43.83	233	**78.04**

- Computing Energy Saved (Wh%)
  - Experiments over 5000<N<150000, Similarity between tables: c=75%
  - Energy savings **~78%** per comparison
  - A future "semantic" search engine can either:
    - reduce energy footprint or
    - increase throughput with same footprint

Results - Profiling Semantic Comparator Kernels on the GPU

TEXAS A&M UNIVERSITY



- Profiling Semantic Kernels
  - Data copy from CPU to GPU (Phase A) ceases to be a bottleneck for N>5k
  - Extracting Scalar coefficients (Phase D) becomes a bottleneck
  - Computing Hash Functions, Insertion into a Bloom Filter (Phases B, C) computationally negligible

Results - Throughput Improvement

Table Size (N)	CPU Throughput (comparisons / s)	GPU Throughput (comparisons / s)	Improvement
5k	53996.91	173097.43	3.20
10k	13458.19	46725.69	3.47
50k	499.40	2025.81	4.05
100k	121.39	510.85	4.20
150k	53.94	228.12	4.22

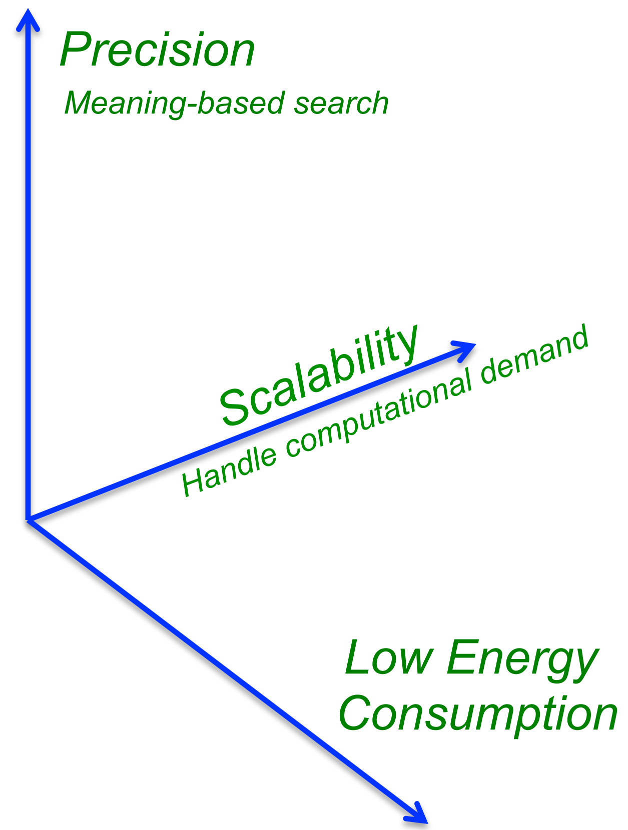
- Improvement in Throughput
  - Ran experiments with randomly varying similarity between tables for given N
  - Throughput was defined as the inverse of the averaged execution time for a given N
  - GPU throughput improvement is higher for larger values of N
  - For smaller values of N, the overhead of data transfer from CPU to GPU dominates

Overview

- Introduction
- Current v/s future technologies
- Key steps in computation
- Description of architecture
- Experimental Setup & Results
- Conclusion

Conclusion

- Semantic search requires introduction of fine-grained parallelism at compute nodes
- Search Engine Precision
  - Use Tensor Method for meaning representation
- Search Engine Scalability
  - Handle explosive growth in coefficient tables within compute nodes
  - Leverage off-the-shelf hardware like GPU's as co-processors
- Search Engine Energy Consumption
  - GPU based semantic comparator has extraordinary energy efficiency
- We have designed GPU based co-processor that provides 4x speedup and 78% energy saving over a traditional CPU for semantic search



Thank You

Q&A

Optimizing a Semantic Comparator using CUDA-enabled Graphics Hardware

Appendix

(Extra Slides)

Comparison with prior art

Characteristics	Traditional Microprocessors	GPGPU	ASIC
Time/Cycles	Worst performing	Medium	24 cycles @ realizable clock frequency
Energy Savings	Worst performing	Moderately high	Very High
Adoption Cost	Low	Intermediate	High fabrication, development, integration costs. IO issues not addressed
Overall characterization	Low speed, Low Cost	Balanced Cost and Speed	High Speed, high cost

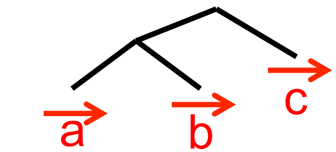
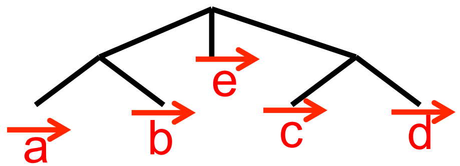
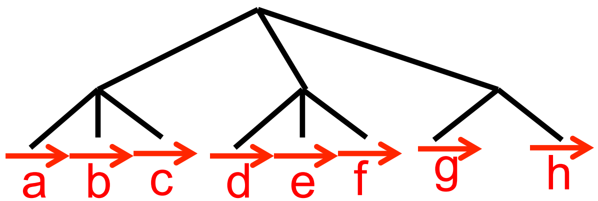
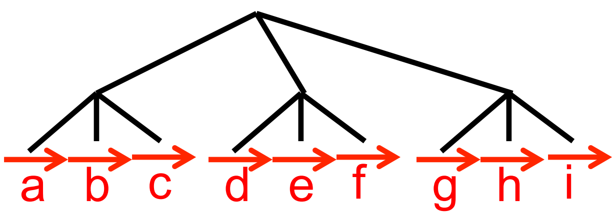
Future Work

- Memory I/O Issues
  - Transmit only hashed dataset to GPU
    - Will reduce dataset from Nx40x2 to Nx8x2 bytes per tables (5 times)
  - Transmit only one Hash instead of two to GPU
    - Compute the second set of hashes in the GPU from the first
    - Will reduce dataset from Nx8x2 to Nx8x1
- Can not Call one kernel from another
  - Control has to pass through the CPU
- Vary GPU Parameters
  - Experimentation with Multiple Grids (In this paper a single grid was used)
  - Further experimentation with varying number of blocks, number of threads per block

References

- S. Mohan, A. Tripathy, A. Biswas, and R. Mahapatra, *"Parallel Processor Core for Semantic Search Engines,"* presented at the Workshop on Large-Scale Parallel Processing (LSPP) to be held at the IEEE International Parallel and Distributed Processing Symposium (IPDPS'11), Anchorage, Alaska, USA, 2011.
- S. Mohan, A. Biswas, A. Tripathy, J. Panigrahy, and R. Mahapatra, *"A parallel architecture for meaning comparison,"* presented at the Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on, Atlanta, GA 2010.
- A. Biswas, S. Mohan, A. Tripathy, J. Panigrahy and R. Mahapatra, "*Semantic Key for Meaning Based Searching*", in 2009 IEEE International Conference on Semantic Computing (ICSC 2009),14-16 September 2009, Berkeley, CA, USA.
- A. Biswas, S. Mohan, and R. Mahapatra, "*Search Co-ordination by Semantic Routed Network*", in 18th International Conference on Computer Communications and Networks, (ICCCN 2009) ,2009, San Francisco, CA, USA.
- A. Biswas, S. Mohan, J. Panigrahy, A. Tripathy, and R. Mahapatra, "*Representation of complex concepts for semantic routed network*," in 10th International Conference on Distributed Computing and Networking, (ICDCN 2009), 2009, Hyderabad, pp 127-138
- A. Biswas, S. Mohan and R. Mahapatra, "*Optimization of Semantic Routing Table*", in 17th International Conference on Computer Communications and Networks, (ICCCN 2008), 2008, US. Virgin Islands

Challenge – Explosive Growth in Number of Terms with Tensor Model

Representation of intent	Number of Index Terms with Vector Method	Number of Index Terms with Tensor Method
	3	7
	5	31
	8	255
	9	511

Current Search Paradigm

- Vector based models
  - Assign weights to keywords
  - Compute similarity using dot product

"The sales manager took the order."

$$D^1 = s^1_{sales}\vec{V}_{sales} + s^1_{manager}\vec{V}_{manager} + s^1_{took}\vec{V}_{took} + s^1_{order}\vec{V}_{order}$$

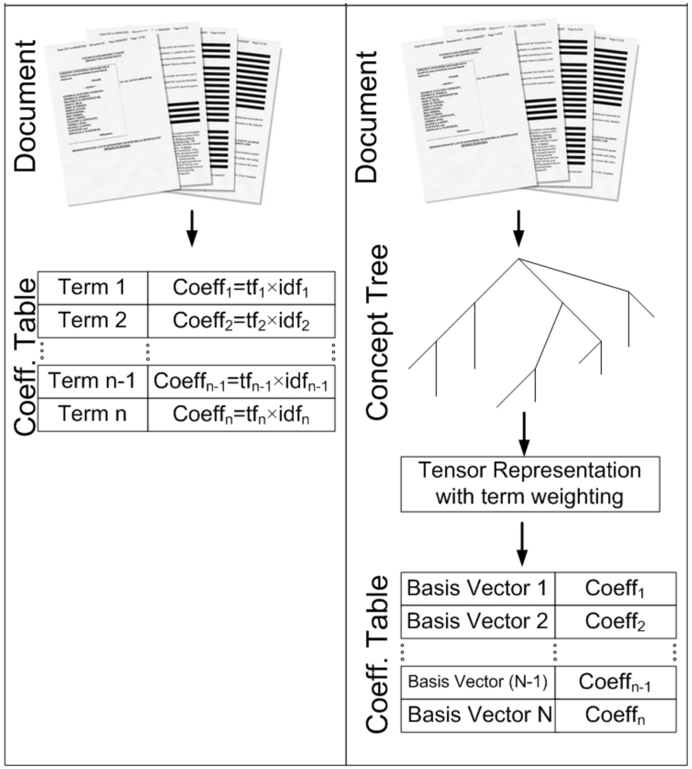
Descriptor representing meaning of the entire text

Basis vector representing the term "sales"

Scalar weight

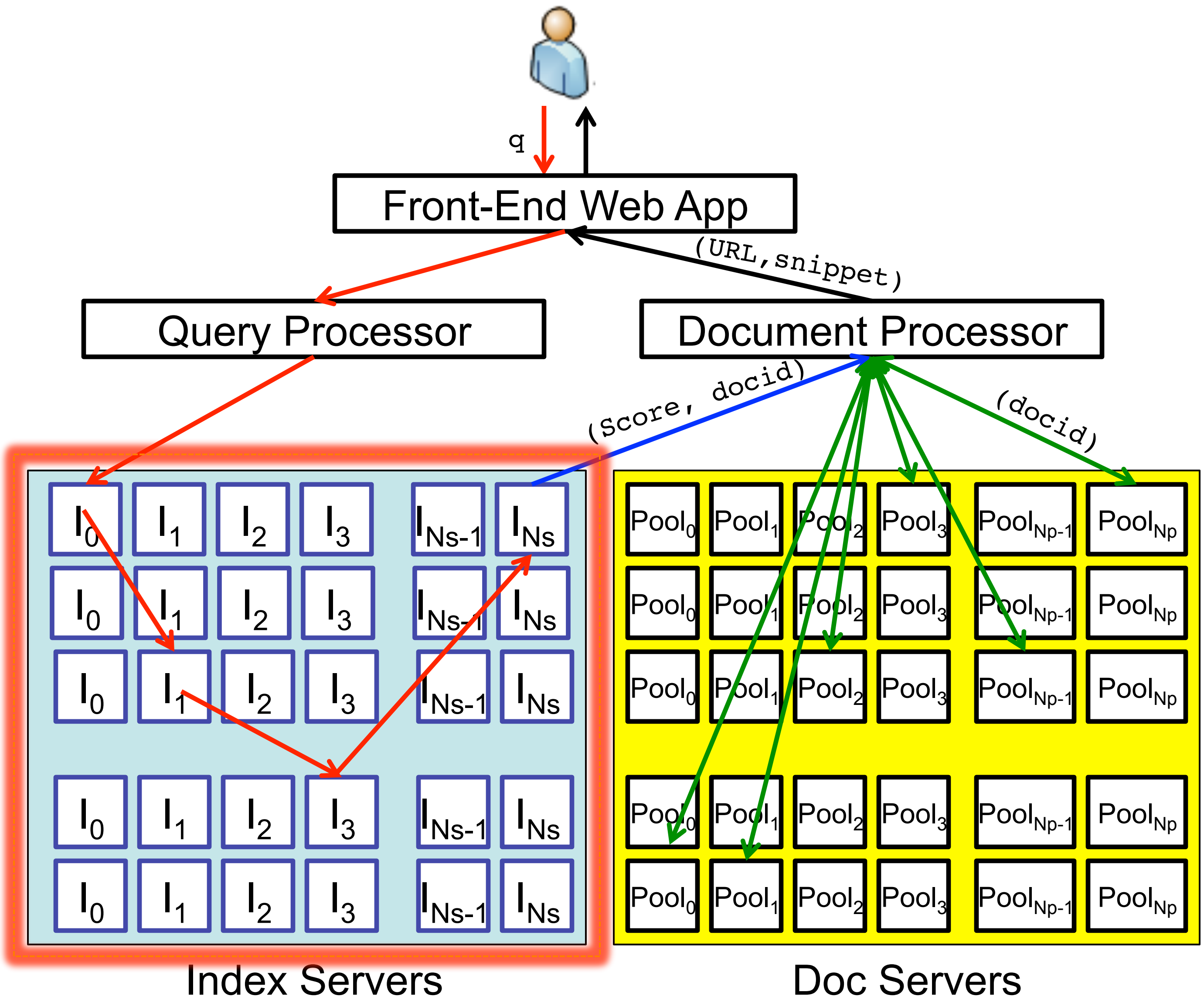
/ coefficient denoting presence of the term

Comparing Current v/s Future Methods



- Creating Coefficient Tables
  - First column shows terms, Second Column shows coefficients
- Tensor Method introduces two additional steps
  - Concept Tree, Tensor Form
  - More computations, but increased precision.

Future – A Semantic Search Engine



- Reorganize the index shards of a search engine
  - Small World Network
  - Reduce Query Rate to <Q/ Ns<<Q
  - Query resolution is guaranteed within a average of 3 hops
  - What is the downside?

Approach

- Use Tensor Based Representation for meaning.
- Meaning Comparison based on dot product of the tensors.

The american man ate indian food

$$s_1 \overrightarrow{\triangleright\triangleright a\, b \triangleleft c \triangleleft} + s_2 \overrightarrow{\triangleright a\, c \triangleleft} + s_3 \overrightarrow{\triangleright b c \triangleleft} +$$

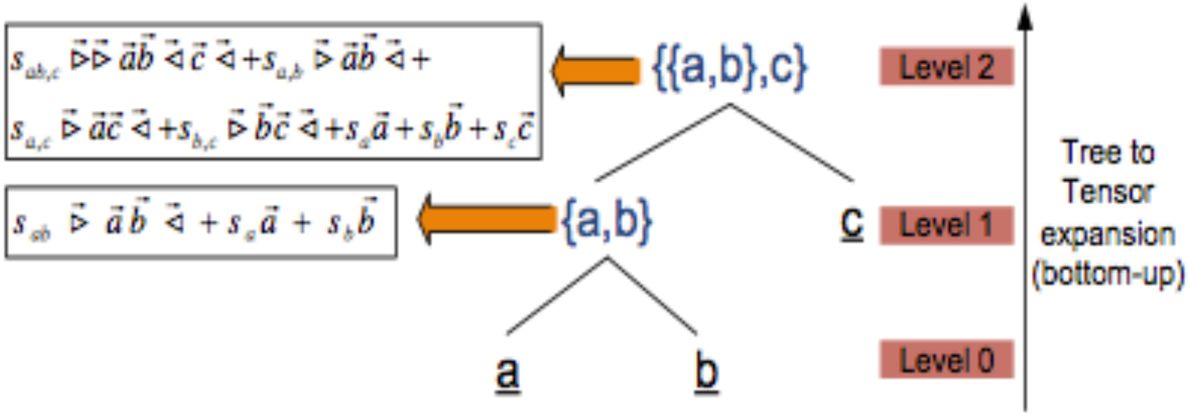
$$s_4 \overrightarrow{\triangleright a\, b \triangleleft} + s_5 \vec{a} + s_6 \vec{b} + s_7 \vec{c} + \cdots$$

Basis vector terms

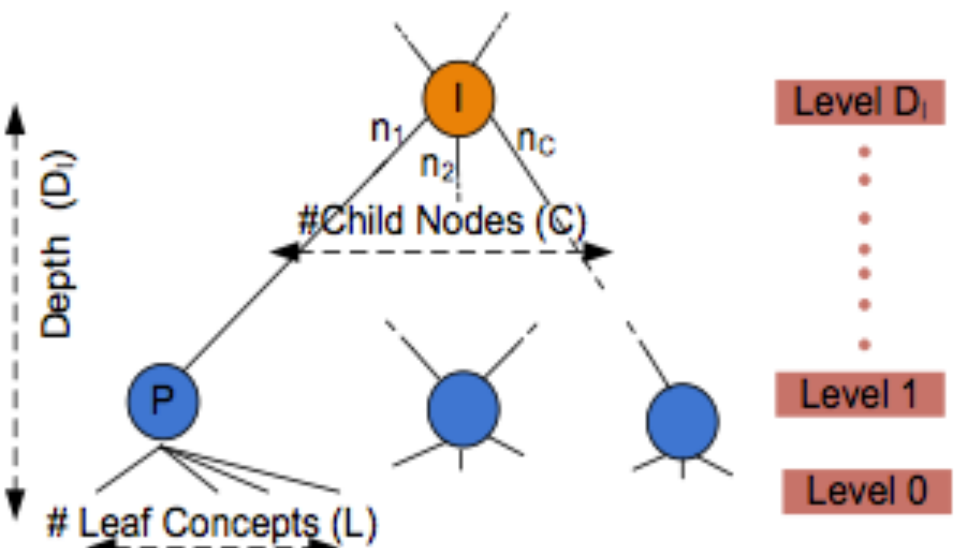
$$\vec{a} = \text{"man"},\ \vec{b} = \text{"american"},\ \vec{c} = \text{"ate"},\ \overrightarrow{\triangleright a\, b \triangleleft} = \text{"}\triangleright \text{ man american } \triangleleft\text{"},$$

$$\overrightarrow{\triangleright b c \triangleleft} = \text{"}\triangleright \text{ american ate } \triangleleft\text{"},\ \overrightarrow{\triangleright\triangleright a\, b \triangleleft c \triangleleft} = \text{"}\triangleright\triangleright \text{ man american } \triangleleft \text{ ate } \triangleleft\text{"},$$

Conversion of a Tree to a Tensor



Example of a specific concept tree



Generic "Concept" tree with C Child Nodes, Depth D, L leaves

- Salient Features
  - Concept Tree: Hierarchical acyclic directed n-ary tree.
  - Lead nodes represent terms whereas the tree describes inter-relationships
- Expansion of Tree
  - Bottoms-up. Make all-possible polyadic combinations
  - Generic Case (Assume):
    - Intermediate Node I
    - "C" Child Nodes
    - One child node "P" contains "L" leaves
    - Number of Terms at D_l due to P will be 2^L-1
    - Each of the C nodes will produce $2^{n1+n2+nC}$ -1 terms

Tensor comparison

(The american man ate indian food)

$$s_1^1\,\overrightarrow{\rhd\rhd ab\lhd c\lhd} + s_2^1\,\overrightarrow{\rhd ac\lhd} + s_3^1\,\overrightarrow{\rhd bc\lhd} + s_4^1\,\overrightarrow{\rhd ab\lhd} + s_5^1\,\vec{a} + s_6^1\,\vec{b} + s_7^1\,\vec{c} + \cdots$$

Tensor (T1)

(The indian man ate american food)

$$s_1^2\,\overrightarrow{\rhd\rhd bc\lhd a\lhd} + s_2^2\,\overrightarrow{\rhd ab\lhd} + s_3^2\,\overrightarrow{\rhd ac\lhd} + s_4^2\,\overrightarrow{\rhd bc\lhd} + s_5^2\,\vec{b} + s_6^2\,\vec{c} + s_7^2\,\vec{a} + \cdots$$

Tensor (T2)

$$\mathrm{Similarity}(\mathrm{T}_1, \mathrm{T}_2) = \mathrm{T}_1 \bullet \mathrm{T}_2 = s_5^1 s_7^2 + s_6^1 s_5^2 + s_7^1 s_6^2 \qquad (<1)$$

1. Identify common basis vectors
2. Multiply scalar coefficients
3. Find sum of all products

Dot product challenge

TEXAS A&M UNIVERSITY

(The american man ate indian food)

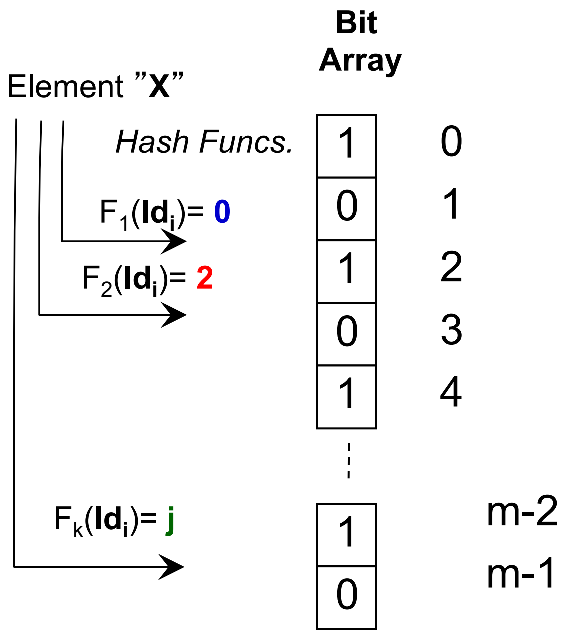
$$s_1 \vec{\rhd}\vec{\rhd}\vec{a}\vec{b}\vec{\lhd}\vec{c}\vec{\lhd} + s_2 \vec{\rhd}\vec{a}\vec{c}\vec{\lhd} + s_3 \vec{\rhd}\vec{b}\vec{c}\vec{\lhd} +$$

$$s_4 \vec{\rhd}\vec{a}\vec{b}\vec{\lhd} + s_5 \vec{a} + s_6 \vec{b} + s_7 \vec{c} + \cdots$$

- When Tensors are large, identification of common basis vectors is time consuming.
- For two Tensors of size n_1, n_2
  - Search is $O(n_1.n_2)$ or $O(n_1.\, \mathit{log}\, n_2)$
- Can we improve upon this?

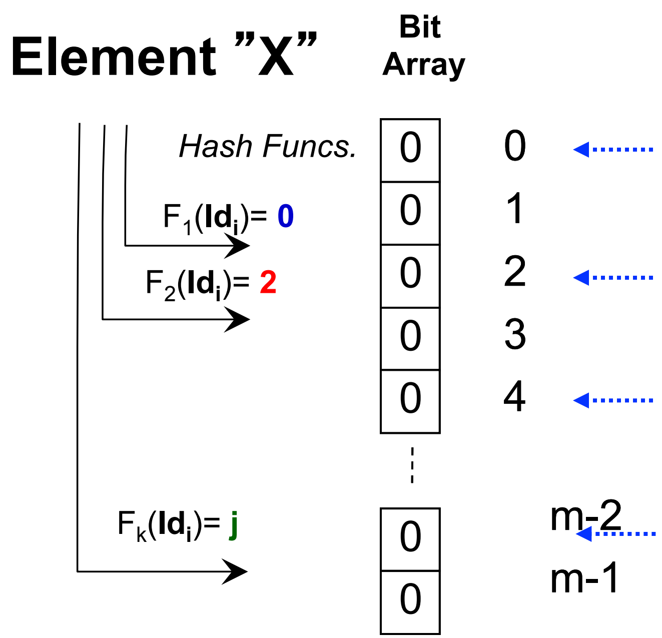
Bloom Filters

- Compact representation of a set.
  - *m* bit long bit vector
  - *k* hash functions



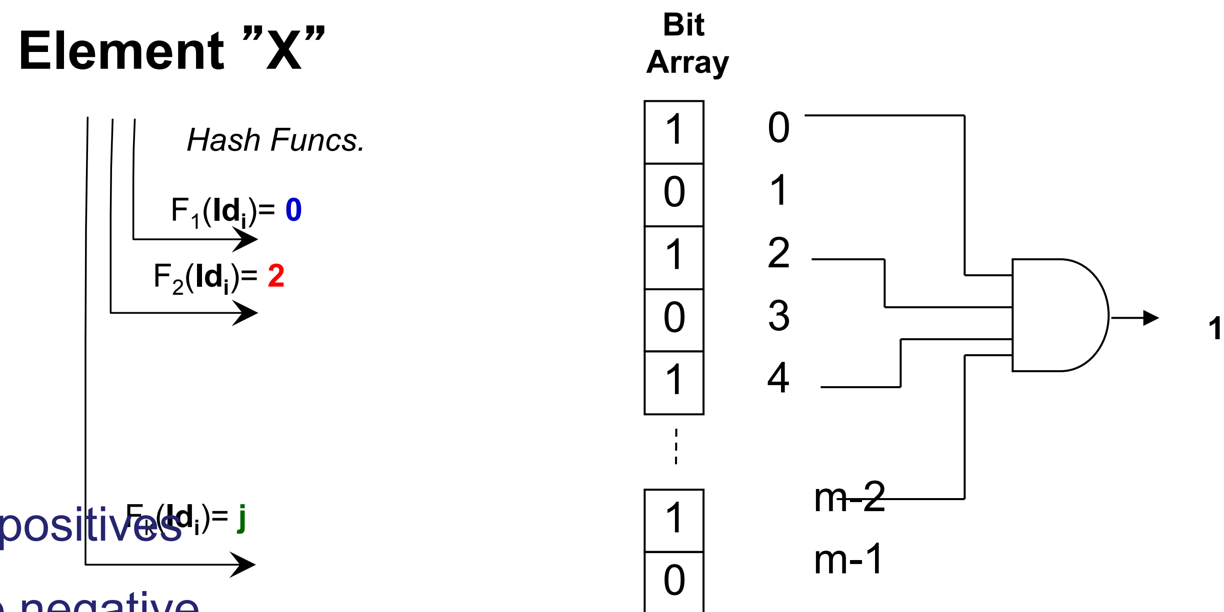
Bloom Filters

- Insertion



Bloom Filters

- Testing for presence (Membership test)



•Can have false positives

•Never have false negative

•False Positive rate can be reduced by choosing large *m* and optimal *k* value.

For n=10^3 elements,

k= 7, m = 10240 bits

Probability of False positive ~ 8x10^{-3}

Data Structure

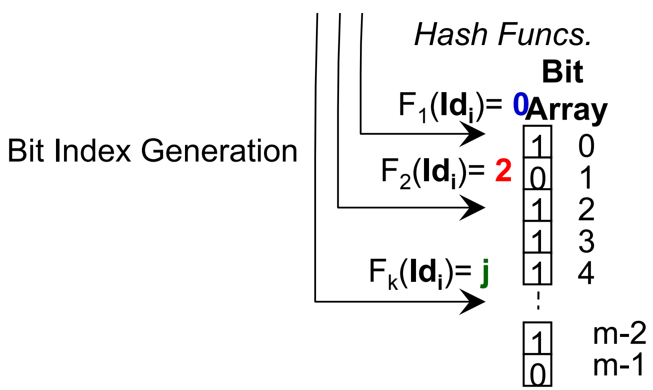
(The american man ate indian food)

$$s_1 \vec{\triangleright}\vec{\triangleright}\vec{a}\vec{b}\vec{\triangleleft}\vec{c}\vec{\triangleleft} + s_2 \vec{\triangleright}\vec{a}\vec{c}\vec{\triangleleft} + s_3 \vec{\triangleright}\vec{b}\vec{c}\vec{\triangleleft} +$$

$$s_4 \vec{\triangleright}\vec{a}\vec{b}\vec{\triangleleft} + s_5 \vec{a} + s_6 \vec{b} + s_7 \vec{c} + \cdots$$

Element "**ac**"

Id_1 = MD5("▹ac◃")



Tensor id	Coeffs	Set of BF bit indices
Id_1	s_1	$\{ x_i : 0 \le x_i \le m \}$
Id_i	$s_i = 0.2$	{ **0**, **2**,…**j** }
Id_n	s_n	{…}

Bloom Filter **Coefficient table**