Parallel LDA with Over-Decomposition

Gordon E. Moon, Aravind Sukumaran-Rajam and P. Sadayappan
Dept. CSE, The Ohio State University

BigDF 2017
December 18, 2017

THE OHIO STATE UNIVERSITY

Top-10 Algorithms in Big-Data/ML

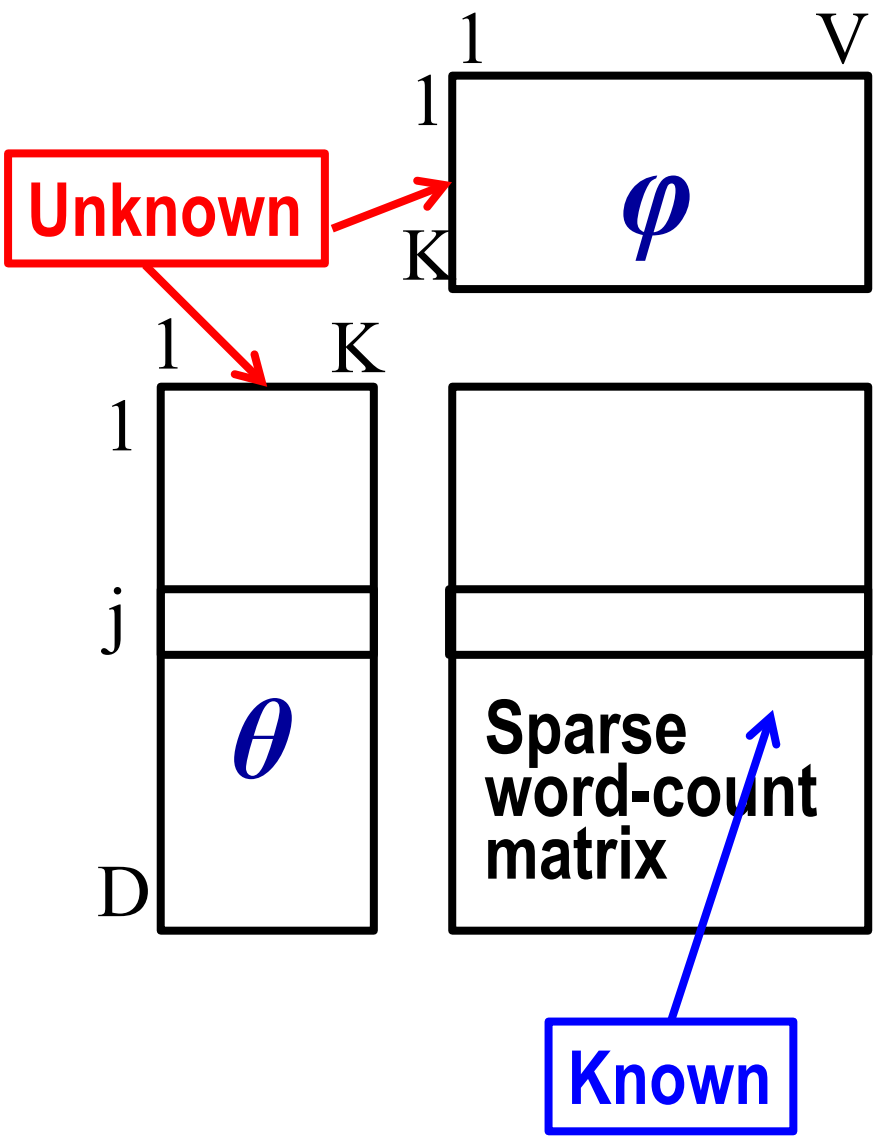
1. Regression (logistic, linear) + Naïve Bayes
2. Support Vector Machines
3. Greedy Clustering (k-Means)
4. Topic Models (Latent Dirichlet Allocation)
5. Collaborative Filtering (Sparse Matrix Factorization)
6. Random Forests
7. Hidden-Markov Models
8. Spectral Clustering
9. Factorization Machines (Regression with Interactions)
10. Multi-layer neural networks
11. Natural Language Parsing

High-performance GPU algorithms are not available for many of these

Source: John Canny

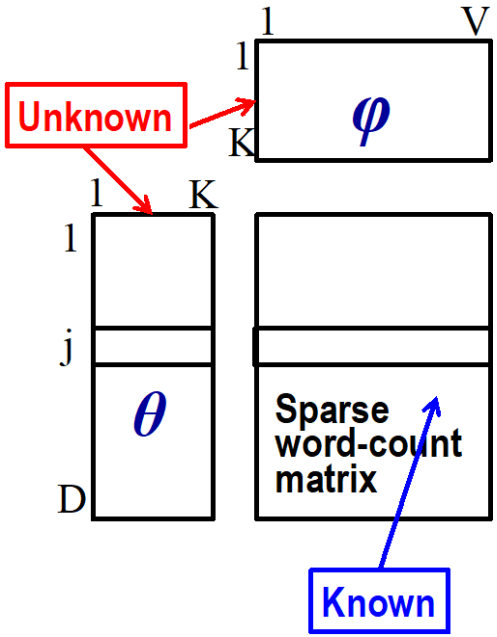
Latent Dirichlet Allocation

- **Latent Dirichlet Allocation (LDA) is a statistical technique for topic modeling**
  - Given *D* documents represented as a collection of words, LDA determines a latent topic distribution for each document
  - Each document *j* of *D* documents is modeled as a random mixture over *K* latent topics → **document-to-topic distribution *θ***
  - Each topic *k* is associated with a multinomial distribution over a vocabulary of *V* distinct words → **topic-to-word distribution *φ***
  - Iteratively improve ***θ*** and ***φ*** until convergence



Latent Dirichlet Allocation

- Three key data structures are read and written as each word token in a document is processed



Iteratively modify via Sampling of distributions

	Actual word token	Word id	Topic id
Document 1	library	1	2
	gym	2	2
	hospital	3	2
Document 2	squash	4	1
	racket	5	1
	squash	4	1
	gym	2	2
Document 3	carrot	6	3
	squash	4	3

DATA: word token list

	Word id			
Document 1	1	2	3	
Document 2	4	5	4	2
Document 3	6	4		

Z: topic assignment list

	Topic id			
Document 1	2	2	2	
Document 2	1	1	1	2
Document 3	3	3		

M: document-to-topic count matrix

	Topic 1	Topic 2	Topic 3
Document 1		3	
Document 2	3	1	
Document 3			2

N: word-to-topic count matrix

	Topic 1	Topic 2	Topic 3
Word 1		1	
Word 2		2	
Word 3		1	
Word 4	2		1
Word 5	1		
Word 6			1

NT: topic count vector

	Count
Topic 1	3
Topic 2	4
Topic 3	2

LDA with Collapsed Gibbs Sampling

- **Collapsed Gibbs Sampling (CGS) is used to estimate posterior distribution over latent variable *z* (word-to-topic assignment)**

	Description
D	Number of documents
V	Number of unique words in the entire corpus
K	Number of topics
x_{ij}	The ith word in document j
z_{ij}	Topic assigned to word x_{ij}
$M_{j\|k}$	2D document-to-topic count matrix: Number of topic k assigned to document j
$N_{x_{ij}\|k}$	2D word-to-topic count matrix: Number of word x_{ij} assigned to topic k
NT_k	1D topic count matrix: Number of topic k in the entire corpus
α, β	Dirichlet prior

M: document-to-topic count matrix

	Topic 1	Topic 2	Topic 3
Document 1		3	
Document 2	3	1	
Document 3			2

N: word-to-topic count matrix

	Topic 1	Topic 2	Topic 3
Word 1		1	
Word 2		2	
Word 3		1	
Word 4	2		1
Word 5	1		
Word 6			1

NT: topic count matrix

	Count
Topic 1	3
Topic 2	4
Topic 3	2

$$P(z_{ij} = k | \mathbf{z}^{\neg ij}, \mathbf{x}, \alpha, \beta) \propto \frac{N^{\neg ij}_{x_{ij}|k} + \beta}{NT^{\neg ij}_{k} + V\beta}(M^{\neg ij}_{j|k} + \alpha)$$

- **Previous efforts at parallelizing LDA have sought to relax CGS and use approximation**
- **In this paper: Parallelize standard Collapsed Gibbs Sampling**

LDA with Collapsed Gibbs Sampling

```
repeat
    for document = 0 to D – 1 do
        L = number of word tokens in document
        for word = 0 to L – 1 do
            current_word_id = DATA[document][word]
            old_topic = Z[document][word]
            decrement M[document][old_topic]
            decrement N[current_word_id][old_topic]
            decrement NT[old_topic]

            Generate new_topic for "word"
            by sampling from current M, N, and NT

            increment M[document][new_topic]
            increment N[current_word_id][new_topic]
            increment NT[new_topic]
            Z[document][word] = new_topic
        end
    end
until convergence
```

DATA: word token list

	Word id			
Document 1	1	2	3	
Document 2	4	5	4	2
Document 3	6	4		

Z: topic assignment list

	Topic id			
Document 1	2	2	2	
Document 2	1	1	1	2
Document 3	3	3		

M: document-to-topic count matrix

	Topic 1	Topic 2	Topic 3
Document 1		3	
Document 2	3	1	
Document 3			2

N: word-to-topic count matrix

	Topic 1	Topic 2	Topic 3
Word 1		1	
Word 2		2	
Word 3		1	
Word 4	2		1
Word 5	1		
Word 6			1

NT: topic count matrix

	Count
Topic 1	3
Topic 2	4
Topic 3	2

Issues in parallelizing LDA

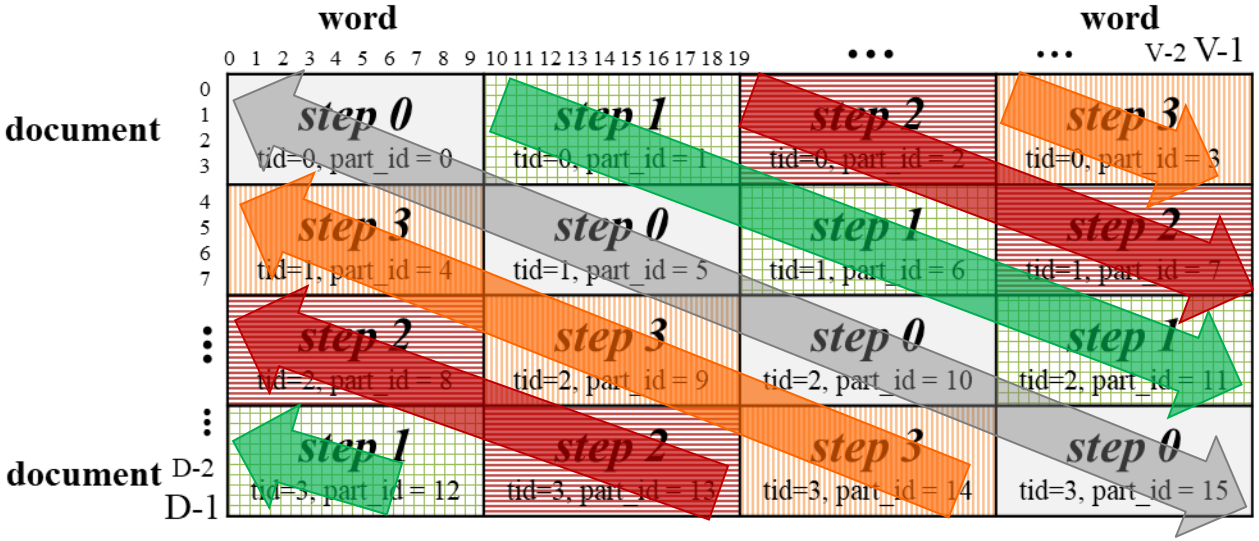
- **Previous approaches**
  - Divide up the documents among parallel thread/processes
    - Common words are present in documents processed by different threads/processes
      - Read/write conflicts on a global word-to-topic matrix
      - ***Atomic*** operation to avoid race condition, but the overheads can be high
  - Maintain local copies of M, N and NT at each thread/process, thereby avoiding atomic operations; merge changes at end of each iteration
    - Sampling process on different threads/processes use distributions from previous iteration => slower convergence

Parallel LDA with Over-Decomposition

- **Approach**
  - Data partitioning
    - 2D tiled approach is used to perform efficient parallel LDA with CGS
  - Load balancing
    - Address the problem of load imbalance using over-decomposition
  - Avoid the need for any atomic operations on updating M or N
    - Additionally, an alternate atomics-free approach introduces an approximation by maintaining local copies for NT

Data Partitioning

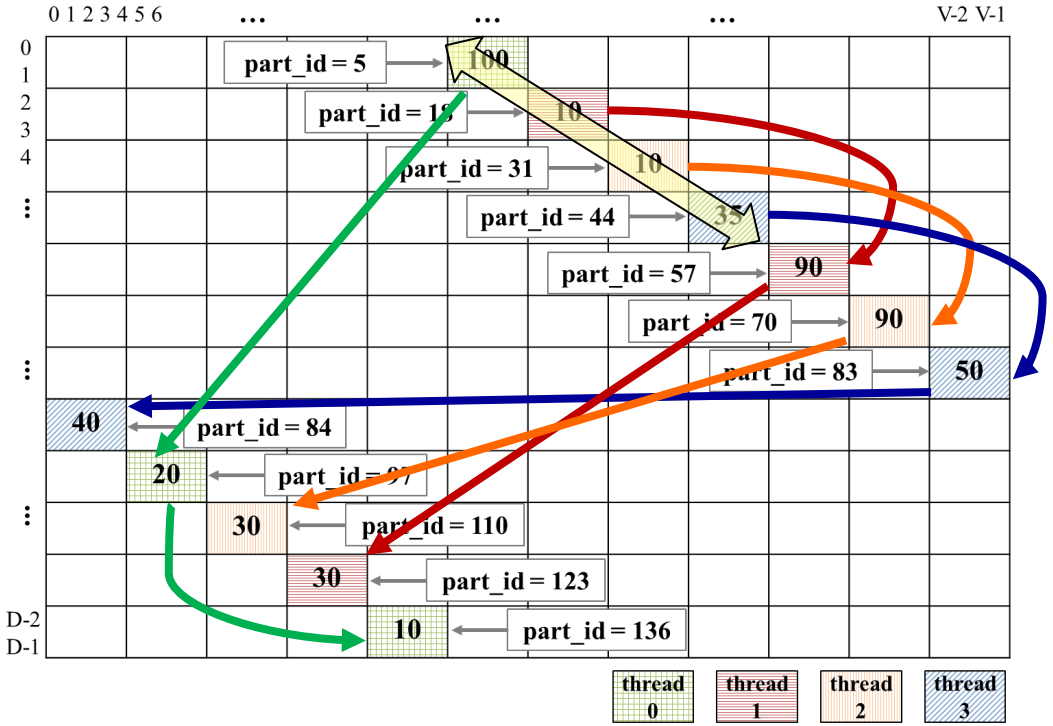
- **Reorganize the sampling over words in documents along diagonals in a 2D tiling of the document-word space**
- **The entire document-word matrix is divided into *P* × *P* partitions**
  - The vocabulary (set of all unique words across all documents) is divided into *P* disjoint partitions
  - Similarly, the documents are also divided into *P* partitions



Document-word matrix: 4 partitions

Load Balancing

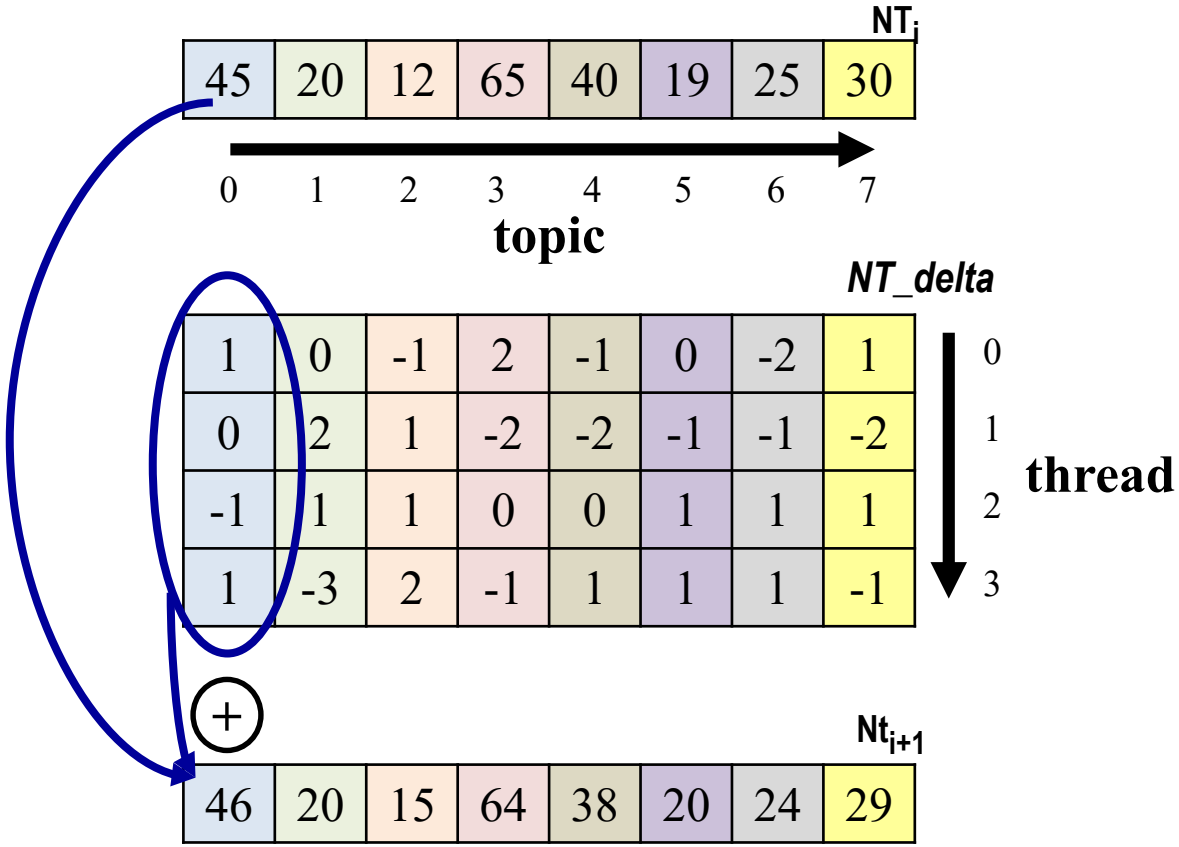
- **The amount of work in each partition is proportional to the number of words**
  - In each 2D partition of the document-word matrix, the number of words tends to be quite non-uniform, with several partitions being empty
- **Load balancing improved by over-decomposition**



Step 5: 12 partitions and 4 threads

Alternate Atomics-free Scheme

- **Using a thread-local delta vector, keep track of the local changes to the NT**
  - Read global state of NT and add the value in its local *NT_delta*
  - Atomic operations are not required, since the updates are made locally



Example: 8 topics and 4 threads

Experimental Evaluation

- **Dataset used**

Dataset	D	W	V
NIPS	1,500	1,932,365	12,375
NYTimes	299,752	99,542,125	101,636

- **Experimental Environment**
  - 8-core multi-processor

Hardware	Intel(R) Xeon(R) CPU E5-2650 v2 @ 2.60GHz (8 cores) 16 GB RAM (1866 MHz)
Software	Red Hat Enterprise Linux Server release 6.7 GCC version 4.9.2

- **Evaluation Metric**
  - The per-word log-likelihood is measured to evaluate the quality of solution
  - The higher the log-likelihood, the better the generalization of the model to unseen data

Experimental Evaluation: Speedup

- **The execution time per iteration depends on the number of threads**

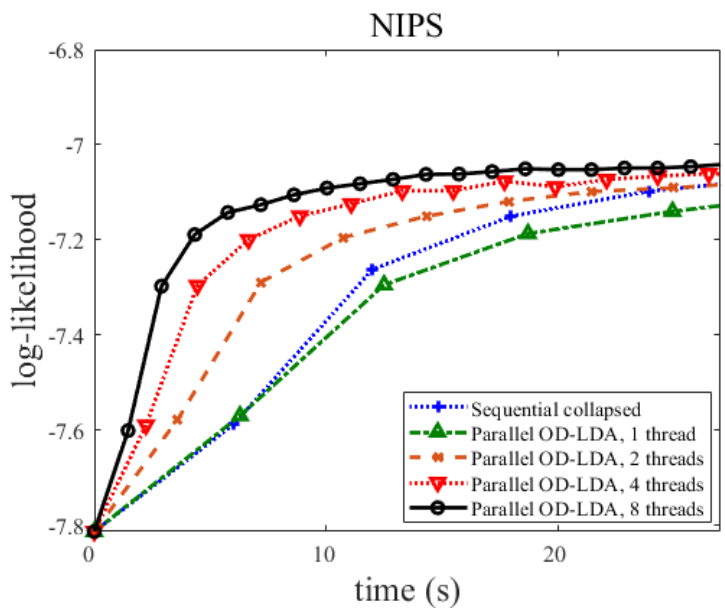
COMPARISON OF THE ELAPSED TIME PER ITERATION ON NIPS AND NYTIMES DATASETS, K=128 AND P=300 ON NIPS AND NYTIMES DATASET.

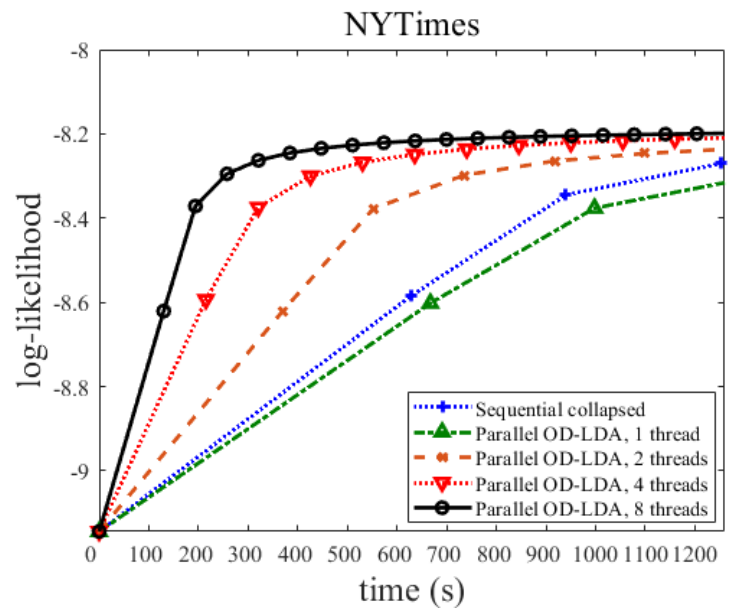
	Number of threads	NIPS Elapsed time per iteration (s)	NYTimes Elapsed time per iteration (s)
Sequential collapsed	None	1.1929	62.6343
Parallel OD-LDA	1	1.2406	66.2232
Parallel OD-LDA	2	0.7132	36.5454
Parallel OD-LDA	4	0.4421	21.0296
Parallel OD-LDA	8	0.2857	12.6711
		4.1 ×	4.9 ×

- Compared to the sequential LDA, with 8 threads, the parallel LDA scheme achieved 4.1 × speedup on the NIPS dataset and 4.9 × speedup on the NYTimes dataset

Experimental Evaluation: Quality of solution

- **Elapsed time vs. Log-likelihood**
  - The new parallel implementation is able to effectively parallelize CGS while maintaining the same log-likelihood

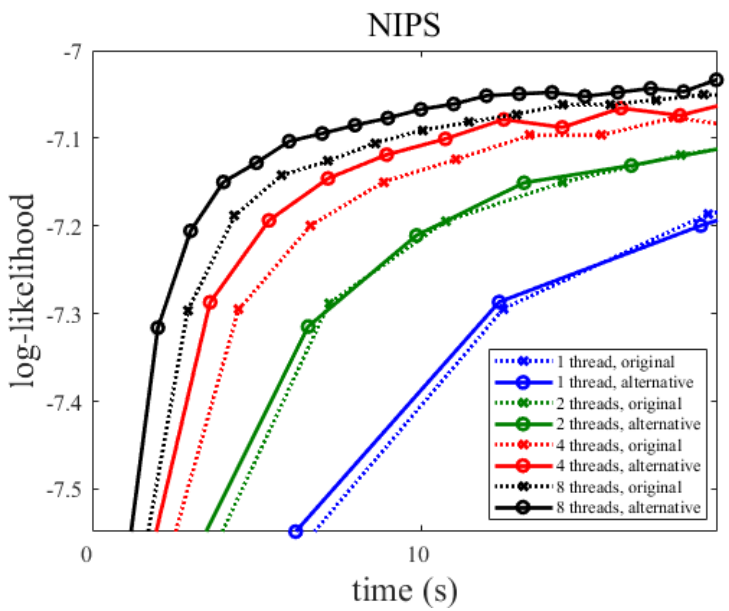


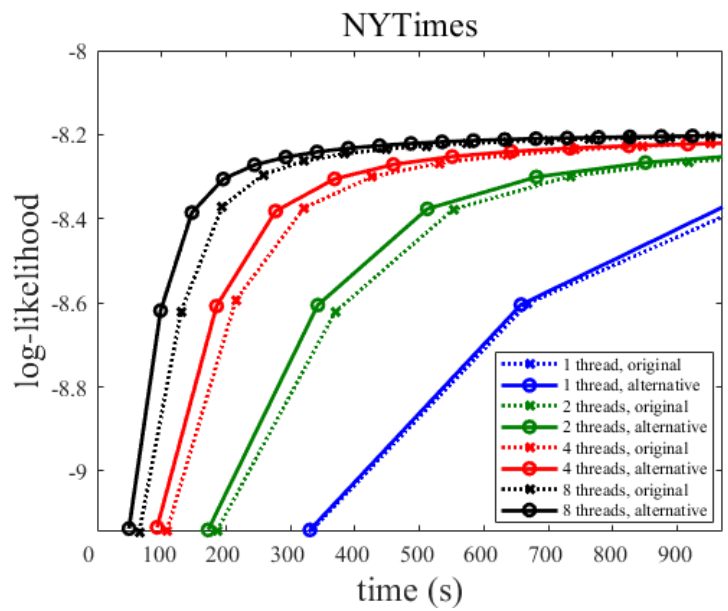


Convergence over time with 1, 2, 4 and 8 threads, on NIPS and NYTimes datasets, K=128. The number of partitions is set to 300 for both NIPS and NYTimes datasets.

Experimental Evaluation: Atomics-free scheme

- **The fully atomics-free scheme (approximation w.r.t. NT) provides additional performance improvement**





original: parallel OD-LDA with atomic operation on only NT

alternative: parallel OD-LDA with fully atomics-free

Summary

- **Developed parallel CGS-LDA that avoids expensive atomic operations on word-to-topic and document-to-topic matrices**
  - Avoid read/write conflicts on a global word-to-topic matrix by 2D tiled data partitioning scheme
  - Use over-decomposition as a means of addressing load imbalance
  - Still requires atomic operations for topic-count vector
- **Devised a fully atomics-free approach that introduces an approximation by maintaining local copies for topic-count vector**
  - Converges faster than a parallel CGS-LDA with atomics
- **Open questions**
  - Can further performance benefits be obtained by further relaxing strictly collapsed Gibbs Sampling? Can convergence be guaranteed?
  - What is the best approach for topic modeling by LDA on GPUs?