# 10-702 Project Report: Parallel LDA, Truth or Dare?

**Aapo Kyrölä**
Department of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
akyrola@andrew.cmu.edu

## Abstract

Latent Dirichlet Allocation (LDA) is a popular topic model, which can be learned using Collapsed Gibbs Sampling. Although Collapsed Gibbs Sampling is inherently sequential, we show that parallel approximation of the original algorithm is able to learn as good models as the exact algorithm. We studied the problem with real datasets, simulated data and statistical analysis.

## 1 Introduction

Machine Learning researchers are faced with exponentially growing amount of data to process for learning tasks. Unfortunately, the performance growth of a single processor chip has stagnated in the recent years and we are forced to resort to parallel computing to improve running time of learning algorithms.

In this project, we study parallelization of learning a Latent Dirichlet Allocation (LDA) [1] model for a large collection of text documents. LDA is a popular probabilistic *topic model* that relates words and documents through a set of topics. Put simply, LDA assumes a generative model, in which each document is associated with a mixture of topics, generated from a Dirichlet prior, and words in the document are generated from these topics. Each word, in turn, has a different probability of being generated from each topic. For example, an article in a NIPS conference might be mostly about classification, but also discusses optimization and image recognition. The words in this article are mostly words common to the three topics. Finally, LDA assumes a bag-of-words simplification and thus does not consider order of words or spatial locality relations.

LDA is an *unsupervised* technique because the topics and word-topic likelihoods are inferred solely from the learning data. One needs to only define the number of topics and values of the hyperparameters of the prior distributions. LDA and other topic models are useful in obtaining a low dimensional representation of a large dataset, and can thus help, for example, in estimating similarities between documents[1] or recognizing "hot" issues (topics) in blogs or scientific articles.

There are two main approaches to learning an LDA model. The original LDA article [1] presents a variational approximation for finding the posterior for the unobserved distributions. This project studies a more recent method that uses Collapsed Gibbs Sampling [3] to directly sample from the posterior. This remarkably simple algorithm is obtained by "collapsing" the posterior by integrating out the latent variables, allowing us to sample from the conditional distributions. Algorithm is discussed in detail in section 2.

Motivation to parallelize LDA model estimation stems from the computational burden. For example, running Gibbs sampling for 1000 iterations (which is a quite common number of iterations to get well mixed samples) on a 1.5 gigabyte dataset and 50 topics on one CPU may take more than 40 hours. If we could run the sampling in parallel, we could save tens of hours in computing time.

[1] Topic models are not limited to text data, but can also used for other purposes such as image classification.

Challenge of parallelizing Collapsed Gibbs Sampling arises from its inherent sequentiality. In this project we study the effect on the quality of learning when the sampling is approximated by sampling for P documents in parallel, and synchronizing sampling state of processors after each processed document. Parallel algorithm is presented in section 3. We show that the approximation does not affect learning quality substantially and indeed, results from parallel learning are practically indistinguishable from sequential, exact computation. In this report, we will present in detail our experiments with real-world datasets (section 4) and analyze by simulation why parallelization works so well (section 5).

## 2 LDA and Collapsed Gibbs Sampling

### 2.1 Latent Dirichlet Allocation

We give here very brief introduction to Latent Dirichlet Allocation (LDA). LDA is based on a generative model, in which documents are assumed to have a multinomial distribution of topics, with a Dirichlet prior. Words in document are generated from topic-specific word distributions (which have another Dirichlet prior), topics are chosen according to the topic distribution of the document.

The total probability of the model is:

$$P(\boldsymbol{W}, \boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{\phi}; \alpha, \beta) = \prod_{k=1}^{K} P(\phi_k; \beta) \prod_{j=1}^{M} P(\theta_j; \alpha) \prod_{t=1}^{N_j} P(Z_{j,t}|\theta_j) P(W_{j,t}|\phi_{Z_{j,t}}) \tag{1}$$

Where $K$ is the number of topics, $M$ number of documents, $N_j$ number of words in document $j$. Bold-font variables denote vector versions.

Explanation of the factors:

1. $P(\phi_k; \beta)$, distribution of words in topic $k$. Distribution is multinomial with Dirichlet prior with uniform parameter $\beta$.
2. $P(\theta_j; \alpha)$, topic distribution for document $j$. Topic distribution is multinomial distribution with Dirichlet prior having uniform parameter $\alpha$.
3. $P(Z_{j,t}|\theta_j)$ topic assignment for *t*th word in document $j$.
4. $P(W_{j,t}|\phi_{Z_{j,t}}) =$ probability of word $t$ in document $j$ given topic $Z_{j,t}$ for the *t*th word in the document.

$\alpha$ and $\beta$ are called *hyperparameters*. Finding appropriate values for them is a matter of art, and in this project we follow [3] and use $\alpha = 50/K, \beta = 0.1$.

To estimate the posterior 1, one can use variational methods [1] or Gibbs sampling, which we describe next. In this project, we are merely interested about estimating $P(\text{word } w|\text{topic } k) \sim \phi_k(w)$.

### 2.2 Collapsed Gibbs Sampling

Gibbs sampling for LDA inference was first introduced in [3]. The idea is to directly sample from the posterior, and use the samples to create an empirical distribution, which is the estimate of the posterior.

*Collapsed* Gibbs sampling means that sampling is done from the posterior where prior distributions $\theta$ and $\phi$ are integrated out. The derivation of the collapsed posterior is not very complicated, but rather long. Interested reader is advised to consult Wikipedia[2], which shows full derivation of the algorithm.

Gibbs sampling algorithm proceeds by sampling a *topic assignment* for each word $w_i$, in each document $j$ from the unnormalized distribution shown in Equation 2. In the equation, $n_{-i,k}^{(w_{j,i})}$ is the number of times word has been assigned to topic $k$, not including the current assignment; $n_{-i,k}^{(\cdot)}$ is

[2] `http://en.wikipedia.org/wiki/Latent_Dirichlet_allocation`

**Algorithm 1**: Collapsed Gibbs Sampler for LDA to estimate word-topic distribution $\phi(w)$. Algorithm has too phases: *burn-in* phase and *sampling* phase. Samples are taken after predetermined intervals $L$ to allow chain to *mix*.

**begin**
- **Burn-in:** For $B$ iterations, sample topics for each word in each document, equation 2..
- **Sampling:**
- **for** $s = 1$ *to* $S$ **do**
  - **for** $l = 1$ *to* $L$ **do**
    - Sample a topic for each word in each document, equation 2.
  - **end**
  - X <- Record topic assignment $z_{j,i}$ for each document $j$ and word $i$.
- **end**
- **Compute posterior:** Use $X$ as an empirical distribution to estimate $\phi$.

**end**

**Algorithm 2**: Parallel Gibbs Sampling for LDA. Each processor samples for disjoint sets of words a time from disjoint set of documents.

**begin**
- **Split** documents $\boldsymbol{D}$ to P (equally sized) sets: $\{\boldsymbol{D}_1, \boldsymbol{D}_2, .., \boldsymbol{D}_P\}$
- **while** *not enough samples* **do**
  - **for** $q = 1$ *to* $P$ **do**
    - **In parallel do** at each processor $p \in \{1...P\}$:
    - **begin**
      - **for** *each document* $d \in \boldsymbol{D}_j$ *where* $j = (p + q) \mod P$ **do**
        - Sample $z_{j,i}$ for words $v$ in document $D_j$ for which $(v \mod P) = p$
        - **Sync point 1:** Synchronize *only* topic-totals between processors.
      - **end**
    - **end**
  - **end**
- **end**

**end**

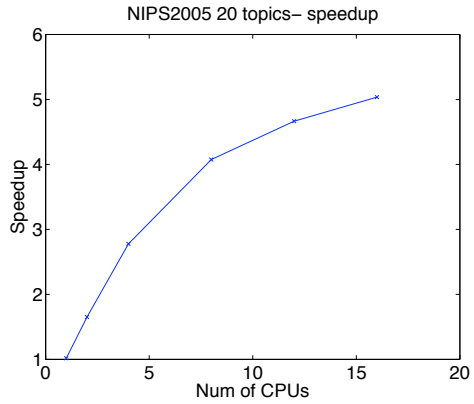
the total number of words assigned to topic $k$, not including the current assignment of the word; $n_{-i,k}^{(d_j)}$ is the number of words assigned to topic $k$ in document $j$ and $n_{-i}^{(d_j)}$ is finally the total number of topic assignments in the document, not including current word assignment. In addition, $W$ is the total size of the vocabulary, and $K$ the number of topics. Intuitively, each probability for assigning a topic is the ratio of times the topic was assigned to the word before, multiplied by the ratio of topics in the document.

$$\mathbb{P}(z_{j,i} = k) \propto \frac{n_{-i,k}^{(w_{j,i})} + \beta}{n_{-i,k}^{(\cdot)} + W\beta} \frac{n_{-i,k}^{(d_j)} + \alpha}{n_{-i}^{(d_j)} + K\alpha} \tag{2}$$

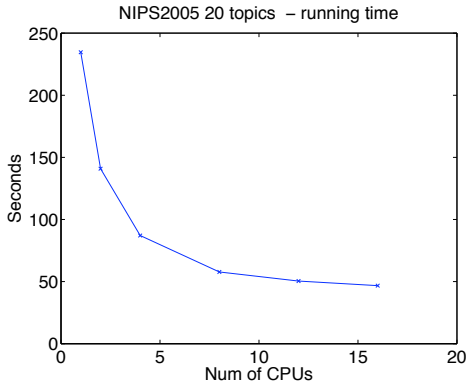
## 3 Parallel Gibbs Sampling Algorithm for LDA

Equation 2 does not readily allow parallelization since each samples depends on previous samples. However, we now present a parallel algorithm that *approximates* sequential Collapsed Gibbs sampling as close as possible. Next section describes experimental results that show that the algorithm indeed works and produces results that are not distinguishable from results produced by the true *sequential algorithm*. This is understandable since Gibbs sampling is a random algorithm and each run will produce different results even when sequentially sampling.

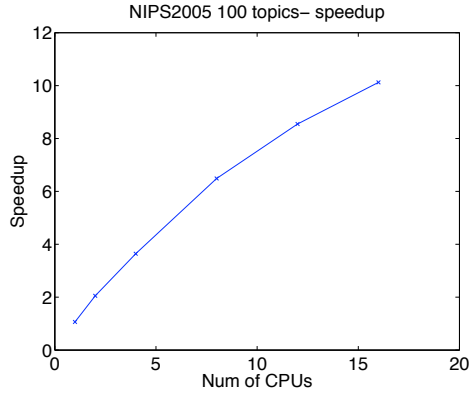
Algorithm 2 is practically equivalent to algorithm presented in [8]. It avoids expensive synchronization of word-counts for each topic since each processor samples for distinct set of words. Each processor samples different document at any time, and total counts are synchronized after each document has been processed by each processor. It is important to note, that between synchronizations,



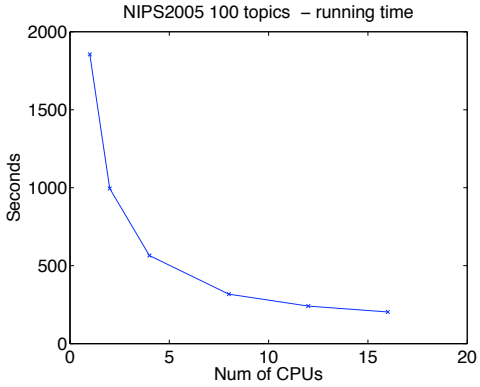

(a) NIPS 20 topics, speedup




(b) NIPS 20 topics, runtime




(c) NIPS 100 topics, speedup




(d) NIPS 100 topics, runtime

only approximately $1/P$ of each document is processed since each processor samples for distinct set of words.

### 3.1 Implementation

Algorithm 2 was implemented in Java by the author. Validity of the results was checked by comparing to *GibbsLDA*[3] software. Software will be available from author's website.

### 3.2 Performance

As the focus of this project was not implementation, we discuss only briefly the parallel performance of the Java implementation. Figure 3.2 shows the speedup (as compared to 1-cpu run) with NIPS 2005 dataset on both 20 and 100 topics. Higher count of topics improves scalability since each sampling takes more time and less time is spent in synchronizing and exchanging data between processors.

All tests were conducted on a 16-core AMD Opteron 2.7 GHz server computer with 64 gigabytes of RAM.

*Remark:* FastLDA [7] is a modified version of the Collapsed Gibbs sampler for LDA, which improves running time significantly by avoiding computing all elements of the sampling distribution (2). However, in this project we did not implement FastLDA because it is more complex and it would not affect the quality analysis for parallel sampling.

[3] `http://gibbslda.sourceforge.net/`

## 4 Statistical comparison of LDA runs

In order to study empirically the effect of parallel relaxation of Collapsed Gibbs Sampling for LDA, we must have a method to compare results of two separate runs of the sampler.

Each run produces a multinomial $P(w|k)$ distribution for each topic $k \in \{1, 2, ..., K\}$. First metric to compare is the prediction power of the distributions, which can be measured using likelihood of data. For topic models most often used metric is *perplexity*, which is inverse of the geometric mean of the per-word likelihood. It is defined as follows:

$$pplex = \exp\Big( - \frac{1}{N} \sum_{d=1}^{D} \sum_{n=1}^{N_d} \log \mathbb{P}(\text{word n | document d}) \Big).$$

Lower perplexity is better and maximum (worst) perplexity is attained when each word is equally likely. We call perplexity *in-sample*, if it was computed on the same data as the model was trained on. All perplexities referenced later are in-sample perplexities, If our aim would had been parameter selection, perplexity should had been computed on a validation-set or test-set.

However, similar perplexity between posteriors is not adequate for determining results are actually similar. Indeed, perplexity is a valid measure only if the topic distribution is checked to be sound by other means. For example, in our work, one broken LDA implementation produced clearly "better " in-sample perplexities than the correct sampler!

In addition to perplexity, we must therefore also evaluate similarity of topic distributions of two different runs. Standard metric for this is to use Kullback-Leibler divergence (KL-divergence). However, topics in LDA are not *identifiable*, i.e topic recognizable as "machine learning" might have index 2 in another run and 74 in another. We follow [6] and use minimum bipartite matching to match topics between two runs. We now discuss this in detail.

### 4.1 Minimum symmetric KL-divergence between two runs

Difference between two probability distributions is often measured with KL-divergence. Standard KL-divergence is not symmetric, so we use a modification called *symmetric KL-divergence*, defined as follows in *bits* for two word distributions with $W$ words:

$$KL^*(P||Q) = \frac{1}{2} \sum_{w=1}^{W} \left( P(w) \log_2 \frac{P(w)}{Q(w)} + Q(w) \log_2 \frac{Q(w)}{P(w)} \right) \tag{3}$$

Here probability of word under topic distribution is simply the ratio of times the word was assigned to the topic $N_w$ and total number of assignments $N$ for the topic, amended with Laplacian smoothing with parameter $\beta$ and size of vocabulary $W$ to avoid probabilities of zero:

$$P(w) = \frac{N_w + \beta}{N + W\beta}$$

Now we have topic distributions from two runs, which we denote as $P^1, P^2, ..., P^K$ and $Q^1, Q^2, ..., Q^K$. To compare two runs, we need to match unidentified distributions which each other and then compute average symmetric KL-divergence between matched topics. Matching can be done using the Hungarian method (see for example [2] p. 247), which finds minimum weighted matching between two parts of a bipartite graph. Edge weights are pairwise symmetric KL-divergences (3). Now let $M$ be a $K \times K$ matrix for which $K(i, j) = 1$ only if $P^i$ matches $Q^j$ and 0 otherwise. Then we can define function for the average KL-divergence between matched topics between two runs:

$$minKL^*(run1||run2) = \frac{1}{K} \sum_{i,j}^{K} KL^*(P_i, Q_j)K(i, j) \tag{4}$$

When comparing two runs, we must remember that due to random nature of sampling, each run will produce slightly different topic distributions. We therefore compare $minKL^*()$ between different sequential runs to $minKL^*()$ between sequential and parallel runs.

### 4.2 Experiments and analysis

#### 4.2.1 NIPS 2005 dataset

First dataset is the collection of articles published in NIPS 2005 conference. This corpus has 206 documents, vocabulary[4] of 18,617 words. We analyzed results with 20 and 100 topics, running sampling in parallel with 4 and 16 cpus.

To get intuition of the scale of symmetric KL-divergence (3), we computed KL-divergences between 20 sequential runs and both random and uniform topic distributions, with 20 topics. In the random distribution, each word has random count between 1 and 100, and for uniform distribution each word had count of 1. In addition, we computed $minKL^*(P_i, P_j)$ (4) between all pairs of the sequential runs. Results are summarized in Table 1.

| Comparison distribution | Mean | Standard deviation | Max | Min |
|---|---|---|---|---|
| Other sequential runs | 1.981 | 0.2313 | 2.736 | 1.371 |
| Random distribution | 4.708 | 0.421 | 6.665 | 3.865 |
| Uniform distribution | 4.327 | 0.419 | 6.186 | 3.545 |

Table 1: Baseline KL-divergences for NIPS 2005, 20 topics

Table 2 compares $minKL^*$ values between sequential and parallel runs. Analysis was done as follows:

1. Run 20 runs of LDA Gibbs Sampling (1000 sampling iterations) with one processor (sequential version) to obtain distributions $\boldsymbol{P}^1, ..., \boldsymbol{P}^{20}$.
2. Run 20 runs with 4 or 16 cpus using the parallel Gibbs sampling to obtain distributions $\boldsymbol{Q}^1, ..., \boldsymbol{Q}^{20}$.
3. Compute $minKL(\boldsymbol{P}^i, \boldsymbol{P}^j), j \neq i$ between all sequential result distributions. Compute mean, min and max. (First row in the table).
4. Compute $minKL(\boldsymbol{P}^i, \boldsymbol{Q}^j)$ between all pairs of results of sequential and parallel runs. Compute mean, min and max. Compute t-test to determine if statistically significantly different compared to minKL-divergences among sequential results. (Second and third row in the table).

| 20 topics | | $minKL^*$ with all sequential runs | | | |
|---|---|---|---|---|---|
| Parallelism | Perplexity (mean $\pm std - dev$) | Mean | Max | Min | p-value |
| 1-cpu runs | $1622.0 \pm 9.6$ | 1.981 | 2.736 | 1.371 | na |
| 4-cpu runs | $1617.1 \pm 10.6$ | 1.958 | 2.768 | 1.195 | 0.138 |
| 16-cpu runs | $1617.1 \pm 7.1$ | 2.014 | 2.808 | 1.256 | 0.072 |

Table 2: NIPS, 20 topics: $minKL$ comparison between sequential and parallel runs. For each configuration, 14 distinct runs were executed.

As one can see, differences between parallel and sequential run are practically same as among sequential runs. The *p-value* in the table is from t-test that compares mean of minKL between pairs of sequential runs and parallel runs to sequential runs. Statistically with 5% confidence, the means are same if p-value is larger than 0.05. This is the case for both parallel sets of runs.

In addition, the divergences are clearly smaller compared to divergences with random or uniform distributions (Table 1), which confirms that the results are essentially indistinguishable. As further

[4]Vocabulary has been cleaned of non-informative stop-words and mathematical symbols.

| 100 topics Parallelism | Perplexity (mean $\pm std - dev$) | $minKL^*$ with all sequential runs Mean | Max | Min | p-value |
|---|---|---|---|---|---|
| 1-cpu runs | $1514.7 \pm 19.0$ | 1.094 | 1.274 | 0.865 | na |
| 4-cpu runs | $1497.9 \pm 15.4$ | 1.053 | 1.202 | 0.885 | 1e-6 |
| 16-cpu runs | $1510.2 \pm 14.8$ | 1.063 | 1.217 | 0.865 | 1e-4 |

Table 3: NIPS: 100 topics, $minKL$ comparison between sequential and parallel runs. For each configuration, seven distinct runs were executed.

evidence, *in-sample* perplexities are close as well. The fact that parallel algorithm produces lower (better) in-sample perplexities hints that parallel algorithm might slightly overfit the data. However, the standard deviation of the perplexity is quite high and it is not possible to make such conclusions from this data.

#### 4.2.2 Enron dataset [4]

Our second dataset is much larger than the first one and consists of emails by Enron employees confiscated by the authorities for the famous fraud investigation. The corpus has c. 520,000 documents with vocabulary of about 180,000 words. Each document has only 123 words in average since most emails are rather brief. Thus synchronization interval is much shorter with our algorithm (unfortunately we did not have time to try increasing the interval) and we obtained only speedup of 4x on 16 cpus. Size of the compressed dataset is 1.1 gigabytes and running 1000 iterations on 1 cpu takes over 45 hours on our computer. Number of topics was 50 and we tested only on 1 cpu and 16 cpus due to long running time of the sampling.

As with the first dataset, we computed symmetric KL-divergences between each sequential run (12 runs), and between a random and uniform topic distributions. Results are summarized in Table 4. We can see that random KL-divergences are over 3 times larger than divergences between different sequential runs.

| Comparison distribution | Mean | Standard deviation | Max | Min |
|---|---|---|---|---|
| Other sequential runs | 2.607 | 0.224 | 3.031 | 1.9762 |
| Random distribution | 8.8699 | 0.9458 | 14.547 | 6.856 |
| Uniform distribution | 8.486 | 0.9345 | 11.219 | 6.508 |

Table 4: Baseline KL-divergences for Enron, 50 topics

Table 5 shows minKL-divergences between parallel and sequential runs. Divergences between sequential and parallel runs are not statistically significantly different with divergences between sequential runs.

| 50 topics Parallelism | Perplexity (mean $\pm std - dev$) | $minKL^*$ with all sequential runs Mean | Max | Min | p-value |
|---|---|---|---|---|---|
| 1-cpu runs | $1997 \pm 3.9$ | 2.607 | 3.031 | 1.976 | na |
| 16-cpu runs | $1998 \pm 3.2$ | 2.531 | 3.236 | 1.886 | 0.139 |

Table 5: Enron, 50 topics: $minKL$ comparison between sequential and parallel runs: 12 sequential runs and five 16-cpu runs.

## 5 Analyzing Parallel Collapsed Gibbs Sampling - Why Does it Work?

Parallel Collapsed Gibbs Sampling for LDA has been experimentally shown to work reliably in many studies.

However, the theory - why does it work - is less well understood. Some intuition is given in [5], which discusses a distributed memory version of the parallel LDA. Authors interpret LDA as an

"approximation to stochastic descent in the space of [topic] assignment variables **z**". In parallel setting, between synchronizations, each processor computes its own ascent direction on the likelihood surface, independently of others, after which the changes are combined in synchronization step. Authors argue that if the surface is locally convex or concave, one expects the combined direction to be accurate as well, but in saddle points behavior will break down. However, the authors continue, "saddle points are 1) unstable and 2) rare", due to the posterior typically being highly peaked (as a result of using Dirichlet priors, see discussion below).

We intuitively agree with this explanation, but certainly a more rigorous analysis would be helpful. In this section we analyze the problem probabilistically, using simulation studies, to show that the approximated posterior in parallel setting is not distinguishable from the true posterior. Unfortunately the posterior is complicated to study analytically, so we fall short of presenting practical bounds on the accuracy.

### 5.1 Simulating approximated posterior

In the Gibbs sampling algorithm, the (collapsed) posterior is estimated by directly sampling from it. In the parallel algorithm, each processor samples for disjoint sets of word tokens and no document is sampled simultaneously by multiple processors. Therefore, in the multinomial distribution for topic assignment (Equation 5), only term $n^{(\cdot)}_{-i,k}$ (in bold), the number of assignments to topic $k$, differs between sequential and parallel settings directly. This value is synchronized by processors after each document (of which they sample approximately for 1/P of the tokens), so the possible error in the value is limited by the synchronization intervals. Therefore, there is a tradeoff between synchronization frequency (which requires atomic updates) and magnitude of deviation from the true posterior.

$$\mathbb{P}(z_{j,i} = k) \propto \frac{n^{(w_{j,i})}_{-i,k} + \beta}{\mathbf{n}^{(\cdot)}_{-i,k} + W\beta} \frac{n^{(d_j)}_{-i,k} + \alpha}{n^{(d_j)}_{-i} + K\alpha} \tag{5}$$

To study the differences in the true and approximated posterior, we use inductive method. We assume that at the beginning of each parallel cycle, the counts used in the posterior are accurate and show that the posterior deviates so little from the true posterior, that this assumption is reasonable on next iteration as well. In essence, it suffices to show that statistically the samples generated by parallel processors are not distinguishable from samples generated by sequential, true, sampler.

#### 5.1.1 Sequential algorithm

To compare approximated posterior used in parallel sampling, we need true posterior of a sequential sampler to compare to. As any sampling order is permissible for a Gibbs sampler, i.e the resulting Markov chain is ergodic independent of the sampling order[5], we choose a sequential sampling algorithm that is most closely resembles the parallel sampler. Algorithm 3 mimics the parallel sampler by sampling for a word token from P documents a time in round-robin order.

#### 5.1.2 Simulation model

We study the difference between approximated posterior and true posterior by repeating following randomized experiment large number of times and computing statistics. The model simulates the state of the posterior *just before synchronization*, when the deviation from true posterior would be the largest in expectation.

| **List of parameters**<br>Symbol | Description |
|---|---|
| $N_j$ | Number of words in document (assumed same for all documents). |
| $N_{total}$ | Total number of words in the training corpus. |
| $W$ | Number of unique word tokens in the training corpus. |
| $K$ | Number of topics. |

[5]However, sampling order may affect mixing speed of the chain, but in our tests, there was no significant difference

**Algorithm 3**: Sequential Gibbs Sampling for LDA. To mimic the parallel algorithm, sequential algorithm (any sequential Gibbs sampling order is correct) samples from P documents a time in a round-robin fashion.

$N_d \leftarrow$ number of documents
$Nw_d \leftarrow$ number of words in document (assume all have same amount)
**begin**
  **Split** documents $\boldsymbol{D}$ to P (equally sized) sets: $\{\boldsymbol{D}_1, \boldsymbol{D}_2, .., \boldsymbol{D}_P\}$
  **while** *not enough samples* **do**
    **for** $t = 1$ *to* $N_d/P$ **do**
      **for** $q = 1$ *to* $P$ **do**
        **for** $v = 1$ *to* $Nw_d$*/P* **do**
          **for** $p = 1$ *to* $P$ **do**
            **for** $t^{th}$ *document* $d \in \boldsymbol{D_p}$ **do**
              Sample $z_{j,i}$ for $v^{th}$ word $v$ in document $d$ for which $(v \mod P) = q + p$
            **end**
          **end**
        **end**
      **end**
    **end**
  **end**
**end**

**Simulate posterior for sampling token $v$ in document $j$ with P parallel processors:**

1. Sample difference in topic distribution $\{\triangle n_k^{seq}\}_{k=1}^K$ by computing difference of two samples from uniform multinomials. This random vector represents the topic assignments *that sequential algorithm 3 would had sampled* before sampling the same word token in same document. The state of the chain affects the change in total number of assignments: in *burn-in*, when all tokens have not been assigned to topics, total number increases. After burn-in, the sum of differences equals zero, since if topic is changed for a word in document, the total number of assignments does not change.
2. Sample the number of times word $v$ appears in the training corpus: $N_v \sim \mathcal{N}(N_{total}/W, N_j/100)$.
3. Sample topic assignment distribution for word $v$ (in the whole chain): $n_{-i,k}^v \sim Multi(N_v, 1/K)$.
4. Sample topic assignment distribution over whole corpus $n_{-i,k}^{(\cdot)} \sim Multi(N_{total}, 1/K)$.
5. Sample topic distribution $\pi(j)$ for document $j$ from Dirichlet prior, according to LDA model, $\pi(j) \sim Dirichlet(\mathbf{1^K}\alpha, 1)$.
6. Sample topic assignments for document $n_{-i,k}^{(d_j)} \sim \pi(N_j, j)$.
7. Compute sequential $P^{seq}(k)$ and parallel $Q^{par}$ versions of the topic distribution, shown in Table 6. Only difference is in the denominator, which includes the $\triangle n_k^{seq}$ term.

Then we analyze statistically the differences between the distributions by computing the KL-divergence and sampling from them.

- Compute KL-divergence $d_{kl}^i = KL(P^{seq}||Q^{par})$. KL-divergence measures difference between two probability distributions.
- Sample $N_j$ samples from the multinomial distributions: $n^{par} \sim Q^{par}(N_j), n_1^{seq} \sim P^{seq}, n_2^{seq}(N_j) \sim P^{seq}(N_j)$.
- Compute $d_{ps}^i = \|n^{par} - n_1^{seq}\|_1, d_{ss}^i = \|n_1^{seq} - n_2^{seq}\|_1$, i.e the sum of absolute values of differences between sample vectors. Comparing statistics of $d_{ps}$ and $d_{ss}$ allows us to analyze whether deviation of samples between true and approximate distribution is different than between two samples from the same sequential distribution.

| $P(z_{j,i} = k \mid \cdot, v)$ | |
|---|---|
| Sequential (true) | Parallel (approximate) |
| $\frac{n^{v}_{-i,k}+\beta}{n^{(\cdot)}_{-i,k}+\triangle n^{seq}_{k}+W\beta} \frac{n^{(d_j)}_{-i,k}+\alpha}{N_j+K\alpha}$ | $\frac{n^{v}_{-i,k}+\beta}{n^{(\cdot)}_{-i,k}+W\beta} \frac{n^{(d_j)}_{-i,k}+\alpha}{N_j+K\alpha}$ |

Table 6: Simulated unnormalized sampling posteriors.

- Record $d^i_{kl}, d^i_{ps}, d^i_{ss}$.

### 5.1.3 Simulation results

Table 7 contains results from simulation that was done with parameters equivalent to which were used for processing the NIPS 2005 dataset. Simulation was done for varying stages of the sampling: first column is the percentage of whole corpus that has been assigned a topic. As expected, in the burn-in phase (any value less than 100%) differences between true and approximate posterior are larger than in the sampling phase (100 %). However, in all stages the differences are very small, and the variability of multiple samples from $P^{seq}$ and between $P^{seq}$ and $Q^{par}$ is equivalent. Therefore, there is no way to distinguish between samples from the two distributions. Please refer to the previous subsection for definitions of the variables. Since the value for KL-divergence is not very informative, third column shows how much means of two normal distributions with same variance would differ for equivalent KL-divergence. Last column is the p-value for a t-test that compares means between $d_{ps}$ and $d_{ss}$ with a one-tailed test, with null hypothesis that the means are same. Apart from the last row, the differences are not statistically significant. It is important to note, that this experiment simulates the sampling of the last word before synchronization, when the discrepancies are highest. Real discrepancies will be lower in average.

| Words assigned | Max $KL(P^{seq}\|\|Q^{par})$ | Comparative $KL(\mathcal{N}\|\|N(x,1))$ | Max $d_{ps}$ | Max $d_{ss}$ | Mean $d_{ps}$ | Mean $d_{ss}$ | T-test p-value $(d_{ps}, d_{ss})$ |
|---|---|---|---|---|---|---|---|
| 0% | $1.20e-04$ | $1.55e-02$ | 98 | 82 | 44.24 | 44.03 | 0.095 |
| 1% | $1.16e-04$ | $1.52e-02$ | 82 | 88 | 44.45 | 44.44 | 0.474 |
| 5% | $8.14e-05$ | $1.28e-02$ | 88 | 82 | 44.01 | 44.34 | 0.975 |
| 10% | $4.27e-05$ | $9.25e-03$ | 84 | 96 | 44.17 | 44.30 | 0.780 |
| 20% | $2.46e-05$ | $7.01e-03$ | 86 | 90 | 44.37 | 44.23 | 0.199 |
| 50% | $4.14e-06$ | $2.88e-03$ | 86 | 90 | 44.68 | 44.66 | 0.448 |
| 100% | $7.64e-07$ | $1.24e-03$ | 86 | 88 | 45.65 | 45.35 | $0.039*$ |

Table 7: Simulation results for comparing true and approximated posteriors $P^{seq}$ and $Q^{par}$ by varying stage of sampling (first column). Parameters mimic the NIPS 2005 dataset: $\alpha = K/50 = 2.5, \beta = 0.1, W = 18617, K = 20, N_{total} = 416,000, N_j = 2021, P = 16$. Number of samples = 5,000 for each row.

Table 8 studies the effect of varying synchronization interval. As expected, longer interval results in higher deviation from the true posterior. However, the deviation grows very slowly and is still not recognizable with synchronization interval of 20,000 words, which corresponds to circa ten NIPS 2005 articles.

Table 9 shows the effect of varying number of topics to the deviation of the parallel sampled posterior from the true posterior. We follow [3] and set hyperparameter $\alpha = K/50$ for each simulation. We can see that the KL-divergence grows when number of topics increases. However, the KL-divergences are not comparable because the distributions have different number of multinomial parameters. Other values do not reveal any tangible intuition for the effect of topic count. This is perhaps explained by two opposing forces: on the first hand, increasing number of topics decreases chance that documents have similar topic distributions and which decreases the discrepancy

| Sync interval, words ($N_j$) | Max $KL(P^{seq}\|\|Q^{par})$ | Comparative $KL(\mathcal{N}\|\|N(x,1))$ | Max $d_{ps}$ | Max $d_{ss}$ | Mean $d_{ps}$ | Mean $d_{ss}$ | T-test p-value $(d_{ps}, d_{ss})$ |
|---|---|---|---|---|---|---|---|
| 100 | $4.70e-08$ | $3.07e-04$ | 14 | 14 | 8.49 | 8.51 | 0.729 |
| 200 | $7.88e-08$ | $3.97e-04$ | 24 | 24 | 12.63 | 12.72 | 0.953 |
| 500 | $2.01e-07$ | $6.34e-04$ | 42 | 40 | 21.33 | 21.22 | 0.099 |
| 1000 | $4.34e-07$ | $9.31e-04$ | 62 | 62 | 30.87 | 31.03 | 0.912 |
| 5000 | $1.88e-06$ | $1.94e-03$ | 140 | 136 | 74.35 | 74.41 | 0.591 |
| 10000 | $3.44e-06$ | $2.62e-03$ | 198 | 194 | 108.86 | 109.00 | 0.640 |
| 20000 | $7.29e-06$ | $3.82e-03$ | 310 | 282 | 158.33 | 158.06 | 0.305 |
| 50000 | $1.62e-05$ | $5.69e-03$ | 450 | 476 | 256.19 | 256.11 | 0.461 |
| 100000 | $2.94e-05$ | $7.67e-03$ | 646 | 648 | 365.77 | 367.22 | 0.888 |
| 200000 | $5.95e-05$ | $1.09e-02$ | 960 | 940 | 522.20 | 516.99 | $0.001**$ |
| 416521 | $1.44e-04$ | $1.70e-02$ | 1406 | 1318 | 771.72 | 751.11 | $0.000**$ |

Table 8: Simulation results for comparing true and approximated posteriors $P^{seq}$ and $Q^{par}$ by varying synchronization interval (first column). Parameters mimic the NIPS 2005 dataset: $\alpha = K/50 = 2.5, \beta = 0.1, W = 18617, K = 20, N_{total} = 416,000, P = 16$. Number of samples = 5,000 for each row.

to true posterior. On the other hand, small number of topics means that individual topics have high assignment counts and thus discrepancies in counts between parallel processors have less effect.

| $K$ | Max $KL(P^{seq}\|\|Q^{par})$ | Comparative $KL(\mathcal{N}\|\|N(x,1))$ | Max $d_{ps}$ | Max $d_{ss}$ | Mean $d_{ps}$ | Mean $d_{ss}$ | T-test p-value $(d_{ps}, d_{ss})$ |
|---|---|---|---|---|---|---|---|
| 2 | $6.11e-08$ | $3.50e-04$ | 48 | 40 | 1.37 | 1.41 | 0.705 |
| 5 | $3.23e-07$ | $8.03e-04$ | 58 | 54 | 8.98 | 9.27 | 0.949 |
| 10 | $5.77e-07$ | $1.07e-03$ | 62 | 66 | 20.90 | 20.70 | 0.196 |
| 15 | $6.92e-07$ | $1.18e-03$ | 70 | 74 | 28.71 | 28.83 | 0.670 |
| 20 | $7.18e-07$ | $1.20e-03$ | 82 | 76 | 35.45 | 35.40 | 0.423 |
| 50 | $1.26e-06$ | $1.58e-03$ | 100 | 100 | 62.59 | 62.73 | 0.705 |
| 100 | $1.40e-06$ | $1.67e-03$ | 138 | 140 | 90.78 | 90.89 | 0.648 |

Table 9: Simulation results for comparing true and approximated posteriors $P^{seq}$ and $Q^{par}$ by varying the number of topics (first column). Parameters mimic the NIPS 2005 dataset: $\alpha = K/50, \beta = 0.1, W = 18617, N_{total} = 416,000, N_j = 2021, P = 16$. Number of samples = 2,000 for each row.

### 5.2 Peakedness of the Dirichlet Prior

Additional insight to explain why parallel sampling works is to look at the properties of the LDA model. As argued in the beginning of this section, the deviation of parallel sampling distribution from the true posterior results in the discrepancy of total topic assignments $\mathbf{n}_{-i,k}^{(\cdot)}$ between parallel samplers. The extent which this causes distributions to deviate depends on how close topic distributions of simultaneously sampled documents are to each other. If each document has very different topic distribution, then the discrepancies on topic assignment counts have only very small effect since document-specific topic counts then determine the modes of the sampling distribution.

Document topic distribution has a $Dirichlet(\alpha)$ prior, where we set $\alpha = 50/K$, where K is the number of topics (following [3]). Table 10 shows - in average - what is the weight of 1-5 largest components of a sample from Dirichlet distribution and how many of the largest components are needed to reach certain quantile of the weight.

Table shows that our Dirichlet prior is quite peaked and puts weight on a small number of components. This further hints why parallel sampling works so well as the chance of sampling two similar documents simultaneously is low.

| K | $\alpha$ | N largest components 1 | 2 | 3 | 4 | 5 | Quantile 0.5 | 0.75 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 25.00 | 0.55 | 1.00 | – | – | – | 1.00 | 2.00 | 2.00 |
| 5 | 10.00 | 0.28 | 0.50 | 0.70 | 0.87 | 1.00 | 2.51 | 3.90 | 4.90 |
| 10 | 5.00 | 0.18 | 0.32 | 0.45 | 0.56 | 0.66 | 3.95 | 6.50 | 8.55 |
| 20 | 2.50 | 0.12 | 0.22 | 0.31 | 0.39 | 0.46 | 6.18 | 10.98 | 15.21 |
| 50 | 1.00 | 0.09 | 0.16 | 0.22 | 0.27 | 0.32 | 10.15 | 19.91 | 30.14 |
| 100 | 0.50 | 0.08 | 0.14 | 0.19 | 0.23 | 0.27 | 13.22 | 27.82 | 45.12 |

Table 10: Peakedness of multinomial distributions sampled from $Dirichlet(\alpha)$ prior. Left side of the table shows the fraction of weight on the 1-5 largest components. Right table shows how many largest components are needed in average to have certain proportion of the weight. Figures are means over 1,000 samples from Dirichlet distribution for each row.

## 6 Conclusions

In this project we presented an efficient parallel algorithm to perform LDA inference using Collapsed Gibbs Sampling. Although the algorithm breaks the sequentiality assumption of the Gibbs sampler, results with real data show that the end result done by parallel algorithm is not distinguishable from the sequential algorithm. Largely this is due to the inexact nature of sampling methods.

We also showed some theoretically inclined analysis on why the parallel approximation works so well. Although we were not able to derive any analytical results, we showed by extensive simulations that in practice the approximated sampling distribution is very close to the true distribution, even if parallel processors synchronize relatively infrequently.

We conclude with answer to the question presented in the subject of this report: Truth or Dare? Fortunately, there is no need to make this trade-off. Daring to use parallelize sampling will not compromise model accuracy noticeably.

## References


[1] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[2] Kenneth Steiglitz Christos H. Papadimitriou. *Combinatorial Optimization*. Dover, 1998.

[3] T.L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences of the United States of America*, 101(Suppl 1):5228, 2004.

[4] B. Klimt and Y. Yang. The enron corpus: A new dataset for email classification research. *Machine Learning: ECML 2004*, pages 217–226, 2004.

[5] D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed inference for latent dirichlet allocation. *Advances in Neural Information Processing Systems*, 20:1081–1088, 2007.

[6] D. Newman, P. Smyth, and M. Steyvers. Scalable Parallel Topic Models. *Journal of Intelligence Community Research and Development*, 2006.

[7] Ian Porteous, David Newman, Alexander Ihler, Arthur Asuncion, Padhraic Smyth, and Max Welling. Fast collapsed gibbs sampling for latent dirichlet allocation. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 569–577, New York, NY, USA, 2008. ACM.

[8] F. Yan, PR Beijing, and Y.A. Qi. Parallel Inference for Latent Dirichlet Allocation on Graphics Processing Units.