Distributed Strategies for Topic Modeling

Prateek Jindal (jindal2@illinois.edu)
Prof. Julia Hockenmaier (juliahmr@illinois.edu)
Prof. L.V. Kale (kale@illinois.edu)
Department of Computer Science, UIUC

MOTIVATION

SERIAL IMPLEMENTATION

- Is quite slow
- Doesn't use the available processors on a machine efficiently

BENEFITS OF PARALLELIZATION

- It is highly efficient
- It can give better clusters
- Large datasets can be processed

Introduction to Topic Models

Latent Dirichlet allocation

Latent Dirichlet allocation (LDA) is a generative probabilistic model of a corpus. The basic idea is that documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. LDA assumes the following generative process for each document **w** in a corpus *D*:

1. Choose $N \sim \text{Poisson}(\xi)$.
2. Choose $\theta \sim \text{Dir}(\alpha)$.
3. For each of the N words w_n:
   (a) Choose a topic $z_n \sim \text{Multinomial}(\theta)$.
   (b) Choose a word w_n from $p(w_n \mid z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

Inference

The key inferential problem that we need to solve in order to use LDA is that of computing the posterior distribution of the hidden variables given a document:

$$p(\theta, \mathbf{z} \mid \mathbf{w}, \alpha, \beta) = \frac{p(\theta, \mathbf{z}, \mathbf{w} \mid \alpha, \beta)}{p(\mathbf{w} \mid \alpha, \beta)}$$

Unfortunately, this distribution is intractable to compute in general. Although the posterior distribution is intractable for exact inference, a wide variety of approximate inference algorithms can be considered for LDA.

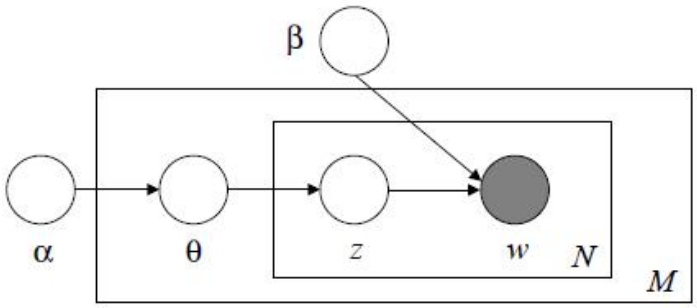
Variational inference

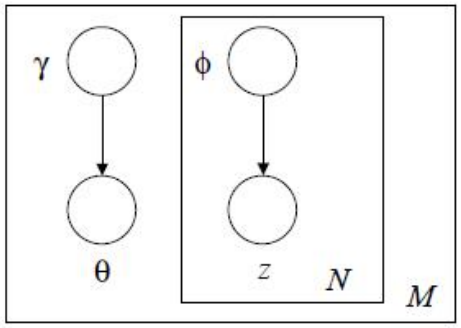
The basic idea of convexity-based variational inference is to make use of Jensen's inequality to obtain an adjustable lower bound on the log likelihood. Essentially, one considers a family of lower bounds, indexed by a set of *variational parameters*. This family is characterized by the following variational distribution:

$$q(\theta, \mathbf{z} \mid \gamma, \phi) = q(\theta \mid \gamma) \prod_{n=1}^{N} q(z_n \mid \phi_n)$$

where the Dirichlet parameter γ and the multinomial parameters ($\phi 1, \ldots, \phi N$) are the free variational parameters. The desideratum of finding a tight lower bound on the log likelihood translates directly into the following optimization problem:

$$(\gamma^*, \phi^*) = \arg\min_{(\gamma,\phi)} \mathrm{D}(q(\theta, \mathbf{z} \mid \gamma, \phi) \parallel p(\theta, \mathbf{z} \mid \mathbf{w}, \alpha, \beta))$$





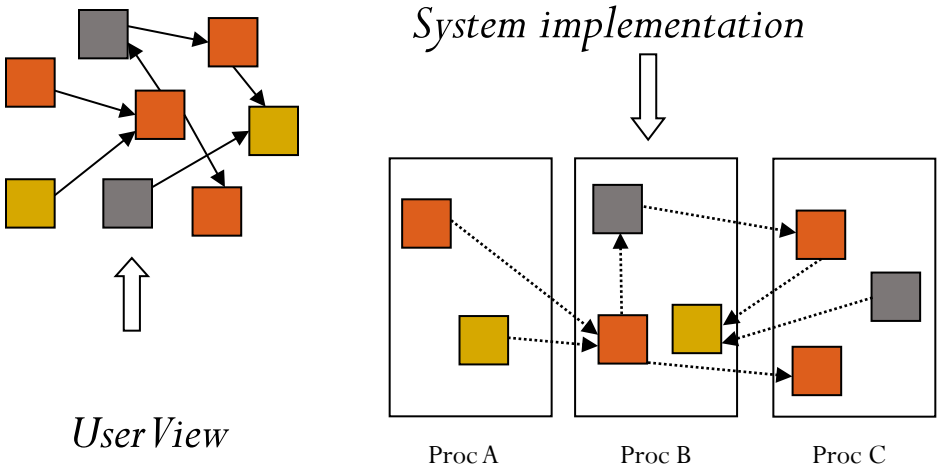
Overview of Charm++

Programmer: [Over] decomposition into virtual processors

Runtime: Assigns VPs to processors

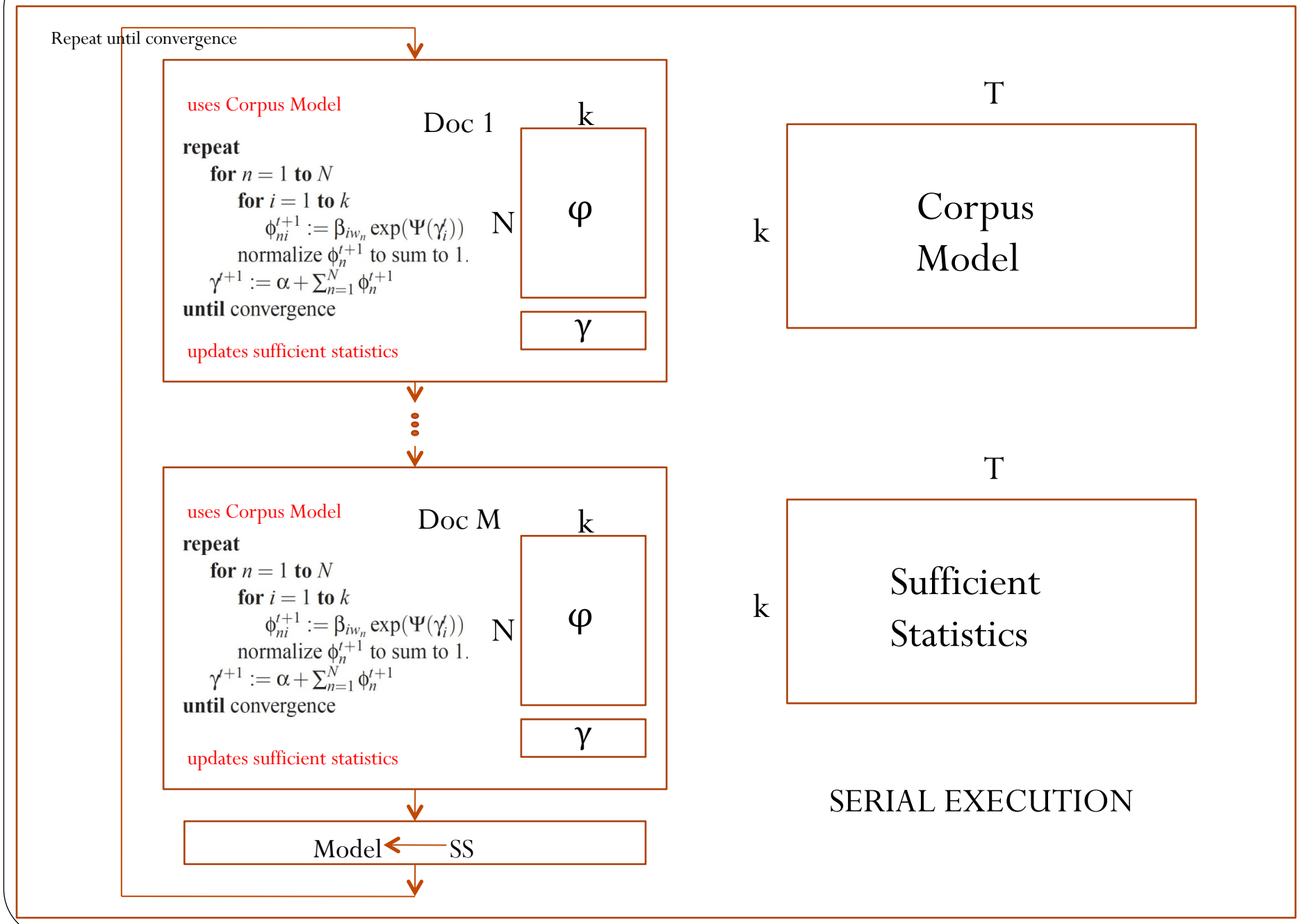
Enables *adaptive runtime strategies*

Implementations: Charm++, AMPI

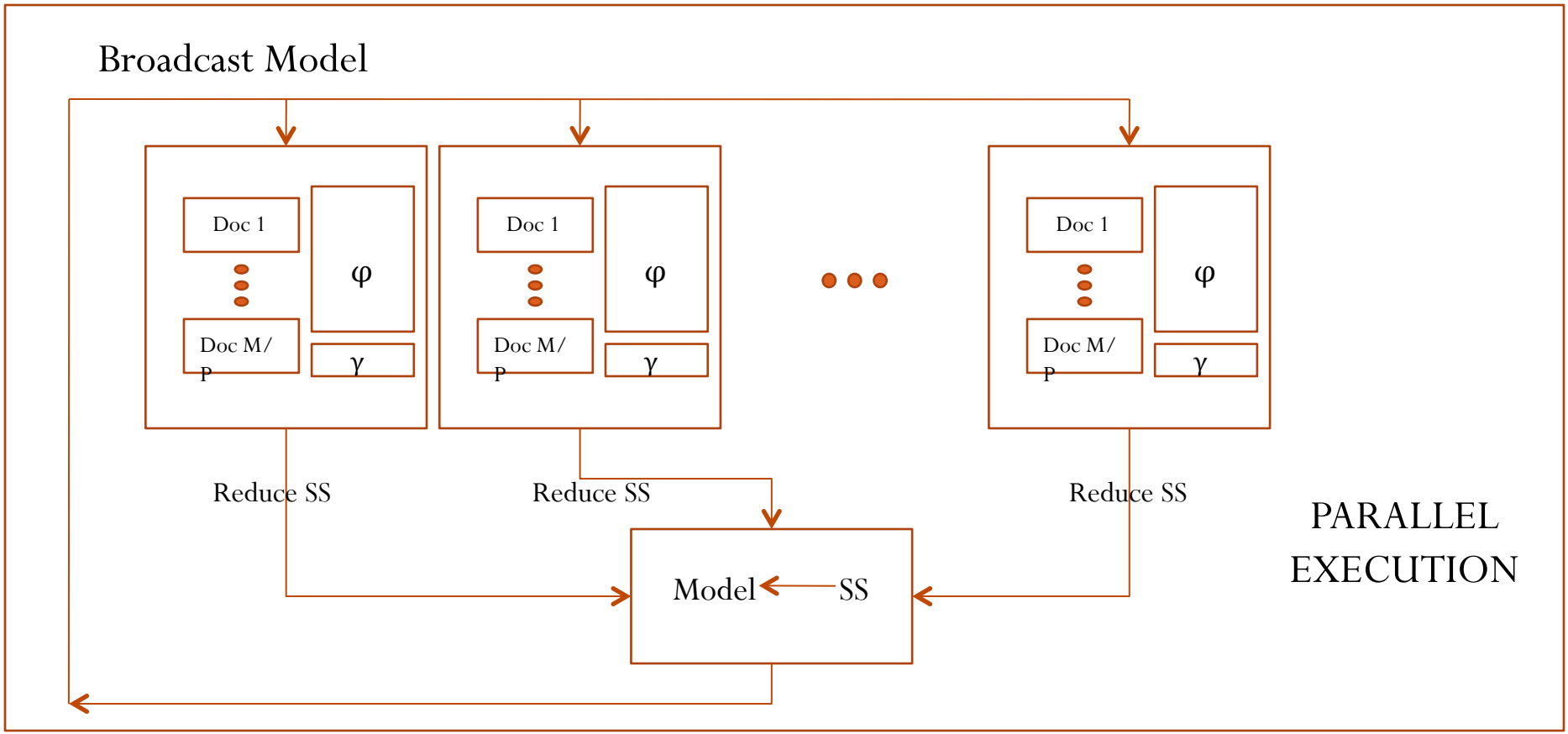


PARALLEL IMPLEMENTATION

1. Split the corpus into multiple parts and assign these parts to different virtual processors.
2. E-step in parameter estimation involves finding the optimal values of the variational parameters for all the documents. Computation of optimal values of variational parameters for different documents is independent and can be done in parallel.
3. However, computation of such optimal values uses the corpus model which is updated after every iteration on a central processor. So, we broadcast the model from the central processor to all the processors in every iteration.
4. After calculating the optimal values of variational parameters, sufficient statistics are calculated for each document. This step is also independent for different documents and can be done in parallel.
5. Once the sufficient statistics for all the documents is obtained, the model is updated from sufficient statistics. Since the model belongs to the whole corpus, we don't parallelize the update step of the model. All the virtual processors send their sufficient statistics to one central processor which carries out the update of the model.
6. In this way, we are able to parallelize the most time consuming part of the EM algorithm which explains the speed-up gained by the parallelization.


Repeat until convergence
uses Corpus Model
Doc 1
k
repeat
for $n = 1$ to N
for $i = 1$ to k
$\phi_{ni}^{t+1} := \beta_{iw_n} \exp(\Psi(\gamma_i^t))$
normalize ϕ_n^{t+1} to sum to 1.
$\gamma^{t+1} := \alpha + \sum_{n=1}^{N} \phi_n^{t+1}$
until convergence
N
φ
γ
updates sufficient statistics
T
k
Corpus Model
uses Corpus Model
Doc M
k
repeat
for $n = 1$ to N
for $i = 1$ to k
$\phi_{ni}^{t+1} := \beta_{iw_n} \exp(\Psi(\gamma_i^t))$
normalize ϕ_n^{t+1} to sum to 1.
$\gamma^{t+1} := \alpha + \sum_{n=1}^{N} \phi_n^{t+1}$
until convergence
N
φ
γ
updates sufficient statistics
T
k
Sufficient Statistics
Model ← SS
SERIAL EXECUTION

Parallel Implementation

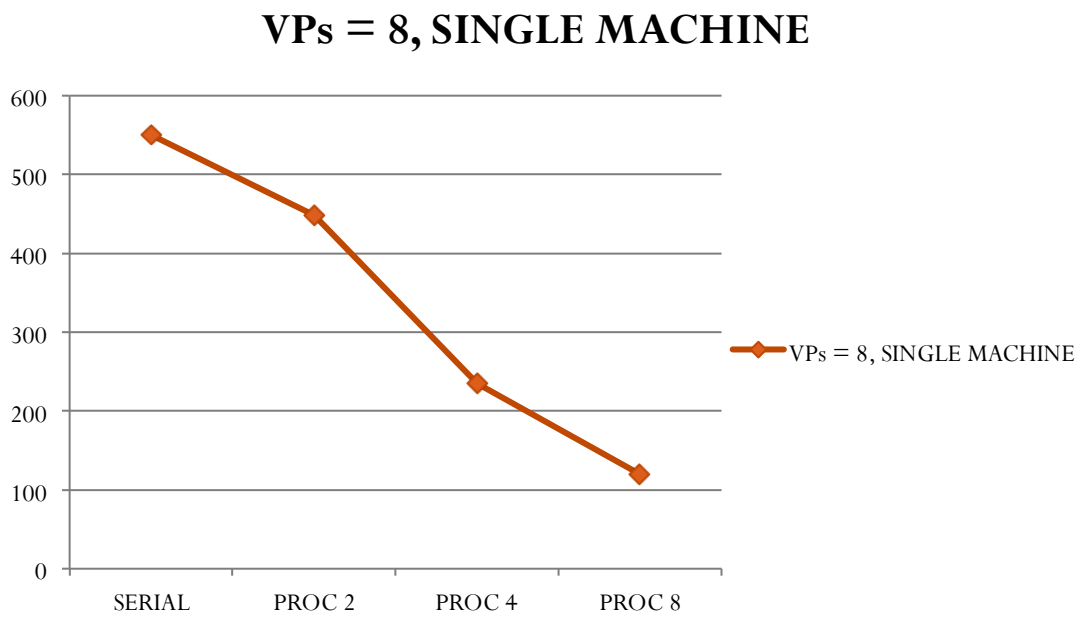


EXPERIMENTAL SETTINGS

- Dataset consisted of 2000 documents from AP
- Total Size of dataset = 2.8 MB
- Total Distinct Terms (T) = 10500
- Number of Topics (k) = 10

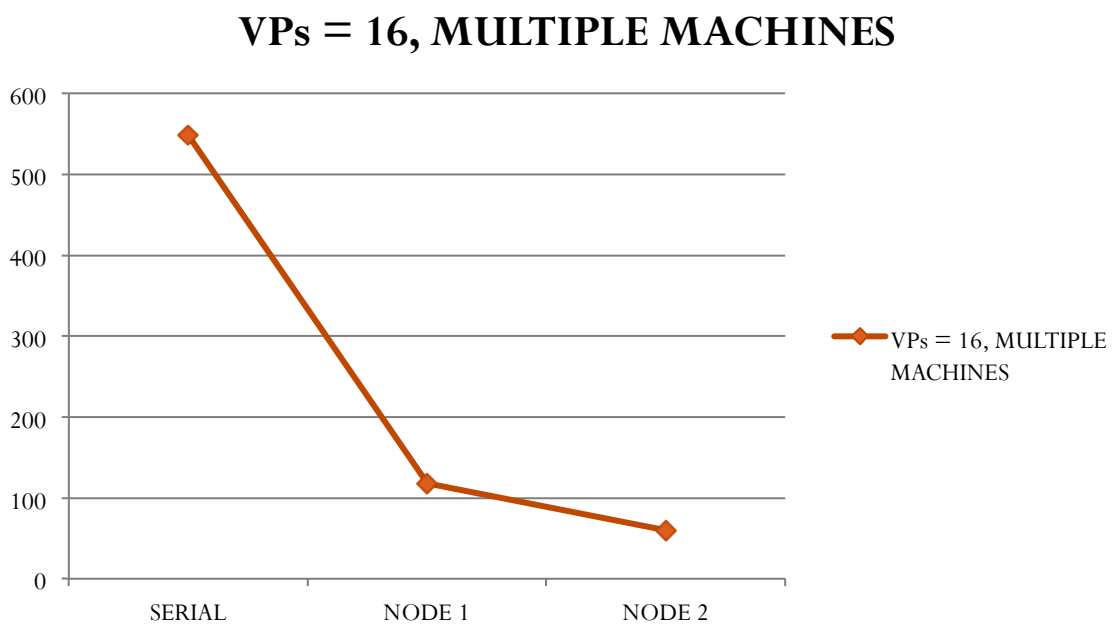
RESULTS

1. The graph below shows the speedup gained by parallelization of topic modeling algorithm on an 8-core desktop. As we increase the number of cores from 2 to 8, the speedup is linear in the number of cores.



RESULTS

2. The graph below shows the speedup gained by parallelization of topic modeling algorithm on multiple 8-core desktops. We see that the introduction of an extra machine reduces the time by half.



FUTURE WORK

- Experiments using larger datasets
- Optimization 1: Executing multiple EM iterations locally before a global reduction
- Optimization 2: Making explicit use of shared memory in multicore