Information Sciences 390 (2017) 15–31

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Towards big topic modeling

JianFeng Yan[a], Jia Zeng[a,c,*], Zhi-Qiang Liu[b], Lu Yang[a], Yang Gao[a]

[a] School of Computer Science and Technology, Soochow University, Suzhou 215006, China
[b] School of Creative Media, City University of Hong Kong, Tat Chee Ave. 83, Kowloon Tong, Hong Kong, China
[c] Huawei Noah's Ark Lab, Hong Kong, China

ARTICLE INFO

Article history:
Received 31 December 2015
Revised 2 November 2016
Accepted 13 December 2016
Available online 14 December 2016

Keywords:
Big topic modeling
Latent Dirichlet allocation
Communication complexity
Multi-processor architecture
Online belief propagation
Power law

ABSTRACT

To solve the big topic modeling problem, we need to reduce both the time and space complexities of batch latent Dirichlet allocation (LDA) algorithms. Although parallel LDA algorithms on multi-processor architectures have low time and space complexities, their communication costs among processors often scale linearly with the vocabulary size and the number of topics, leading to a serious scalability problem. To reduce the communication complexity among processors to achieve improved scalability, we propose a novel communication-efficient parallel topic modeling architecture based on a power law, which consumes orders of magnitude less communication time when the number of topics is large. We combine the proposed communication-efficient parallel architecture with the online belief propagation (OBP) algorithm, referred to as POBP, for big topic modeling tasks. Extensive empirical results confirm that POBP has the following advantages for solving the big topic modeling problem when compared with recent state-of-the-art parallel LDA algorithms on multi-processor architectures: (1) high accuracy, (2) high communication efficiency, (3) high speed, and (4) constant memory usage.

1. Introduction

Probabilistic topic modeling [4,25,31,36] represents a powerful method for data analysis in machine learning and applied statistics. Using topic modeling, we represent collections of documents as a document-word co-occurrence matrix, where each element is the number of word counts in the document. In this paper, we study one of the most successful topic modeling algorithms, latent Dirichlet allocation (LDA) [5], which has been wildly used in many fields such as text mining, computer vision and computational biology [10,16,26,30,35]. From a thematic labeling point of view, LDA assigns the hidden topic labels to explain the observed words in the document-word matrix. We can infer the topic labels from the observed words using LDA by computing the posterior distribution of the hidden variables given the observed variables from their joint probability following the Bayes rule.

In the big data era, we need big topic modeling algorithms that can infer a large number of parameters in LDA from big data. The big document size D, the vocabulary size V and the number of topics K determine the large-scale topic modeling problems [34], which can enhance several real-world industrial applications such as search engine and online advertising systems. In practice, the computation and communication costs of topic modeling are proportional to all three

* Corresponding author.
E-mail addresses: yanjf@suda.edu.cn (J. Yan), j.zeng@ieee.org (J. Zeng), ZQ.LIU@cityu.edu.hk (Z.-Q. Liu), yanglu@suda.edu.cn (L. Yang), gaoyang.suda@gmail.com (Y. Gao).

http://dx.doi.org/10.1016/j.ins.2016.12.014
0020-0255/

elements. Therefore, big topic modeling must reduce both the computation and communication costs when D, V and K become sufficiently large. Big topic modeling algorithms have attracted intensive research interests because big data have become increasingly common in recent years such as in the form of the billions of tweets, images and videos on the web. The applications of big topic modeling include search engine systems with massive search queries, online advertising systems, social strength learning in social networks, network analysis [41], recommendation systems, computer vision [7] and music analysis [15].

However, it is still a difficult challenge to reduce both the time and space complexities of traditional batch LDA algorithms such as variational Bayes (VB) [5], collapsed Gibbs sampling (GS) [13], and belief propagation (BP) [46] for big topic modeling tasks. For example, if we use the batch BP [46] to extract 10,000 topics from the PUBMED data set, which contains 8.2 million documents [24], the memory required to store all the documents and the LDA parameters is approximately 36 TB, and the time consumption for 200 iterations is approximately 3 months on a single processor. Therefore, both the time and space costs are unaffordable in many real-world applications.

In this paper, we focus on reducing the communication complexity in MPA for big topic modeling tasks. It is not difficult to combine MPA and MCA to achieve a better parallel architecture to solve big topic modeling problems. To achieve the goal of communication efficiency, we propose a novel MPA based on a power law [23], which is observed to achieve a few orders of magnitude lower communication costs when compared with state-of-the-art parallel LDA algorithms [1,22,24,39,51]. Moreover, we combine this parallel architecture with the state-of-the-art online LDA algorithm OBP [49], referred to as POBP, for big topic modeling tasks. In our experiments, our POBP runs 5–100 times faster, uses a constant memory space, and requires approximately 5–20% of the communication time but achieves 20–65% higher topic modeling accuracy than state-of-the-art parallel LDA algorithms.

The remainder of this paper is organized as follows. Section 2 introduces related work on big topic modeling. Section 3 reviews (1) the online belief propagation (OBP) algorithm [49] and (2) the current MPA scheme [22] for big topic modeling. Section 4 presents our solution, POBP, and introduces how to use the power law to significantly reduce the communication complexity in MPA. Section 5 compares the proposed POBP with several state-of-the-art parallel LDA algorithms. Finally, Section 6 draws conclusions and envisions further work.

2. Related work

Recent big topic modeling solutions fall into three categories: (1) fast batch LDA algorithms, (2) online LDA algorithms, and (3) parallel LDA algorithms. Fast batch LDA algorithms focus on improvements in running speed on a single machine, whereas parallel LDA algorithms enhance the running speed by partitioning the documents across processors such that the amount of data on each processor is reduced significantly to reduce the running time. Online LDA algorithms can reduce memory space requirements greatly by running small mini-batches, which run slightly faster than their batch counterparts, at each iteration.

Fast batch LDA algorithms observe the fact that the probability mass of the topic distribution is concentrated on a small set of topics when the number of topics is very large. This sparseness property facilitates fast Gibbs sampling (FGS) [24] and sparse Gibbs sampling (SGS) [40] algorithms. The basic idea is to sample a topic by checking the topics with high concentrated probability mass first. Generally, FGS and SGS run approximately 8–20 times faster than traditional GS [13] when the number of topics is very large. Active belief propagation (ABP) [47] is a sublinear BP algorithm [46] for topic modeling. At each iteration, ABP scans only a subset of topics and documents to achieve a high convergence speed. In practice, ABP is approximately 10–20 times faster than SGS and FGS in terms of convergence and achieves a higher topic modeling accuracy. Despite the high speed on large data sets, anchor word recovery-based topic modeling algorithms [2] scale nonlinearly with the vocabulary size and the number of topics. Although a significant speedup has been achieved, these fast batch LDA algorithms still require a large memory space to store both data and LDA parameters. Moreover, the speedup of fast batch LDA algorithms is limited to within approximately 20, thus restricting their usage in large-scale applications.

Unlike fast batch solutions, online LDA algorithms require only a constant memory space by treating both data and LDA parameters as streams composed of several small mini-batches. After sequentially loading each mini-batch into memory for computation until convergence, we free each mini-batch from memory after one look. In practice, we need to confirm that online algorithms can converge to the local optimum point of LDA's objective function. Within the stochastic optimization framework [6], online variational Bayes (OVB) [14] and online belief propagation (OBP) [49] have been demonstrated to fulfill this goal. Generally, online algorithms are faster than their batch counterparts by a factor of 2 to 5 due to the use of fast local gradient descents. However, online algorithms rarely use powerful parallel architectures to further scale their performances because of high communication costs or serious race conditions [1,37].

Parallel LDA algorithms use widely available parallel architectures to speed-up the topic modeling process. Currently, there are two types of parallel architectures, multi-processor [22,42,43] and multi-core [18,37], where the difference lies in how the memory is used. In the multi-processor architecture (MPA), all processes have separate memory spaces and communicate to synchronize LDA parameters at the end of each iteration. In the multi-core architecture (MCA), all threads share the same memory space, and thus, the race condition is serious. There are three important questions remaining to be addressed in recent parallel LDA algorithms:

1. Accuracy: Can parallel LDA algorithms produce the same results as those of batch counterparts on a single processor?

Table 1
Notations.

$1 \le d \le D$	Document index
$1 \le w \le W$	Word index in vocabulary
$1 \le k \le K$	Topic index
$1 \le m \le M$	Mini-batch index
$1 \le n \le N$	Processor index
$\mathbf{x}_{W\times D} = \{x_{w,d}\}$	Document-word matrix
$\mathbf{z}_{W\times D} = \{z^k_{w,d}\}$	Topic labels for words
$\boldsymbol{\theta}_{K\times D}$	Document-topic distribution
$\boldsymbol{\phi}_{K\times W}$	Topic-word distribution
α, β	Dirichlet hyperparameters

2. Communication cost: How does one reduce the communication cost in MPA?
3. Race condition: How does one alleviate the race condition in MCA?

Almost all parallel GS (PGS) algorithms [1,19,22,24,29,33,37] can yield only an approximate result to that of batch GS [13], whereas parallel VB (PVB) [51] is able to produce exactly the same result as that of batch VB [5]. To alleviate the race conditions on the GPU MCA, a streaming approach is proposed to partition data into several non-conflict data streams in memory [37]. However, this partitioning process may introduce the loading imbalance problem and thus a low parallel efficiency. Concerning MPA, the reduction of the communication cost remains an unsolved problem since the communication cost is often too large to be masked by the computation time in web-scale applications [12]. The experimental results confirm that the communication cost may exceed the computation cost to become the main cost of big topic modeling [19,33]. Therefore, the reduction of communication costs can facilitate obtaining a high scalability performance for parallel LDA.

3. Preliminary algorithms

We briefly review OBP [49] and MPA [22] for big topic modeling. We show that a simple combination of OBP and MPA will cause unaffordable communication costs and thus a poor scalability performance. In light of this problem, we propose a communication-efficient MPA framework to greatly reduce the communication cost. Table 1 summarizes important notations in this paper.

LDA allocates a set of thematic topic labels, $\mathbf{z} = \{z^k_{w,d}\}$, to explain non-zero elements in the document-word co-occurrence matrix $\mathbf{x}_{W\times D} = \{x_{w,d}\}$, where $1 \le w \le W$ denotes the word index in the vocabulary, $1 \le d \le D$ denotes the document index in the corpus, and $1 \le k \le K$ denotes the topic index. Usually, the number of topics K is provided by users. The nonzero element $x_{w,d} \neq 0$ denotes the number of word counts at the index $\{w, d\}$. For each word token $x_{w,d,i} = \{0, 1\}, x_{w,d} = \sum_i x_{w,d,i}$, there is a topic label $z^k_{w,d,i} = \{0, 1\}, \sum_{k=1}^K z^k_{w,d,i} = 1, 1 \le i \le x_{w,d}$. We define the soft topic label for the word index $\{w, d\}$ by $z^k_{w,d} = \sum_{i=1}^{x_{w,d}} z^k_{w,d,i} x_{w,d,i}/x_{w,d}$, which is an average topic labeling configuration over all word tokens at index $\{w, d\}$. The objective of LDA is to maximize the joint probability $p(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\phi}|\alpha, \beta)$, where $\boldsymbol{\theta}_{K\times D}$ and $\boldsymbol{\phi}_{K\times W}$ are two non-negative matrices of multinomial parameters for document-topic and topic-word distributions, satisfying $\sum_k \theta_d(k) = 1$ and $\sum_w \phi_w(k) = 1$. Both multinomial matrices are generated by two Dirichlet distributions with hyperparameters α and β. For simplicity, we consider the smoothed LDA with fixed symmetric hyperparameters [13].

3.1. OBP

Online belief propagation (OBP) [49] combines active belief propagation (ABP) [48] with a stochastic gradient descent framework [6]. OBP partitions the document-word matrix $\mathbf{x}_{W\times D}$ into mini-batches $x^m_{w,d}, 1 \le d \le D_m, 1 \le m \le M$. After loading the mth mini-batch into memory, OBP infers the posterior probability, called a message, $\sum_k \mu^m_{w,d}(k) = 1$, $\mu^m_{w,d}(k) = p(z^{k,m}_{w,d,i} = 1|x^m_{w,d,i} = 1, \boldsymbol{\theta}, \boldsymbol{\phi}; \alpha, \beta)$,

$$\mu^m_{w,d}(k) \propto \frac{[\hat{\theta}^m_{-w,d}(k) + \alpha] \times [\hat{\phi}^m_{w,-d}(k) + \beta]}{\hat{\phi}^m_{-(w,d)}(k) + W\beta}, \tag{1}$$

where $\hat{\theta}$ and $\hat{\phi}$ are the *sufficient statistics* for the online LDA model,

$$\hat{\theta}^m_{-w,d}(k) = \sum_{-w} x^m_{w,d}\mu^m_{w,d}(k), \tag{2}$$

$$\hat{\phi}^m_{w,-d}(k) = \hat{\phi}^{m-1}_w(k) + \sum_{-d} x^m_{w,d}\mu^m_{w,d}(k), \tag{3}$$

in which $-w$ and $-d$ denote all word indices except w and all document indices except d, respectively, and $-(w,d)$ denotes all indices except $\{w,d\}$. The multinomial parameters of the document-topic and topic-word distributions $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ can be obtained by normalizing the sufficient statistics $\hat{\boldsymbol{\theta}}$ and $\hat{\boldsymbol{\phi}}$. Each mini-batch is swept for several iterations T_m until the convergence condition is reached. Then, OBP frees from memory the mth mini-batch, the local $\mu^m_{w,d}(k)$ and $\hat{\theta}^m_{-w,d}(k)$. The global topic-word distribution $\phi^m_w(k)$ in memory is later re-used by the next mini-batch. When $\phi^m_w(k)$ is very large, we may also store the entire matrix on a hard disk and load the partial matrix in memory for computation [49].

OBP is an ideal choice for big stream topic modeling on a single-processor platform because of several advantages. First, OBP guarantees convergence to the stationary point of LDA's likelihood function within the online expectation-maximization (EM) framework [8,17,21]. Second, OBP is made memory-efficient by using a disk as the storage extension. OBP's space complexity in memory is proportional to the mini-batch size D_m and the number of topics K. Finally, OBP is built upon the time-efficient ABP algorithm [48], whose time complexity is insensitive to the number of topics K and the number of documents in each mini-batch D_m. However, the communication complexity is intractable if we directly parallelize OBP in MPA for big topic modeling tasks, which will be explained in detail in the next subsection.

3.2. MPA

The MPA scheme has been widely used in many parallel batch LDA algorithms [19,22,24,33,39,51]. Here, we extend this scheme to parallelize online LDA algorithms. The MPA [22] distributes each mini-batch $\mathbf{x}_{W\times D_m}$ of documents over $1 \le n \le N$ processors. The processor n receives approximately $D_{m,n} = D_m/N$ documents. The local $\hat{\boldsymbol{\theta}}^{m,n}_{K\times D_{m,n}}$ can also be distributed into N processors; however, the global $\hat{\boldsymbol{\phi}}^{m,n}_{K\times W}$ have to be shared by N processors since each distributed mini-batch $\mathbf{x}_{W\times D_{m,n}}$ may still cover the entire vocabulary of words. After sweeping each mini-batch $\mathbf{x}_{W\times D_{m,n}}$ at the end of each iteration $1 \le t \le T_m$, the N processors have to communicate and synchronize the global matrix $\hat{\phi}^{m,.,t}_{K\times W}$ from N local matrices $\hat{\phi}^{m,n,t}_{K\times W}$ by

$$\hat{\phi}^{m,.,t}_w(k) = \hat{\phi}^{m,.,t-1}_w(k) + \sum_{n=1}^{N}[\hat{\phi}^{m,n,t}_w(k) - \hat{\phi}^{m,.,t-1}_w(k)]. \tag{4}$$

Then, the synchronized matrix $\hat{\phi}^{m,.,t}_w(k)$ is transferred to each processor to replace $\hat{\phi}^{m,n,t}_w(k)$ for the next mini-batch. Thus, the communication complexity is

$$\text{Communication complexity} \propto NMTKW, \tag{5}$$

where N is the number of processors, M is the number of mini-batches, K is the number of topics, W is the vocabulary size, and $T = \sum_{m=1}^{M} T_m/M$ is the average number of iterations to reach convergence for each mini-batch. For example, suppose that we use 1000 processors to learn $K = 2000$ topics with $T = 100$ from the PUBMED data set [24], with $W = 141,043$ and $M = 500$ mini-batches. The total communication cost is approximately 100 PB (10^{15} bytes) according to (5). Meanwhile, the time complexity of OBP decreases linearly with the number of processors N. Thus, the communication cost will be greater than the computation cost when $N \to \infty$. In this situation, adding more processors will not reduce the total topic modeling time, leading to serious scalability issues. The major reason why MPA worked in previous parallel batch LDA algorithms [19,22,24,33,39] is that the communication cost depends only on the number of batch iterations T' rather than on the number of iterations over mini-batches MT, where $T' \ll MT$ in practice. If $T' = 500$, the parallel batch LDA algorithms only require a 1 PB communication cost in the above example, which is significantly smaller than that of parallel online LDA algorithms. For certain real-world big data streams, the number of mini-batches may diverge to infinity [50], i.e., $M \to \infty$. Thus, the communication cost of parallel online LDA algorithms may become so large as to seriously degrade the parallel efficiency.

Fig. 1 compares the communication costs between parallel batch and online LDA algorithms. Parallel batch LDA algorithms communicate and synchronize $\hat{\boldsymbol{\phi}}_{K\times W}$ at the end of each batch iteration, whereas parallel online algorithms do this at the end of each mini-batch iteration. Generally, the number of batch iterations T' is significantly smaller than the number of mini-batch iterations MT. Thus, the higher communication rate leads to the larger communication cost in parallel online LDA algorithms. Therefore, it is nontrivial to reduce the communication complexity (5) for parallel online LDA algorithms [14,20,32,39,49] to achieve a better scalability performance. Moreover, not all parallel batch LDA algorithms based on MPA have been proven to converge to the local optimum of the LDA's objective function. Typical examples include GS-based parallel algorithms [19,22,24,33,39] in the MPA framework.

4. POBP

In this paper, we propose a communication-efficient MPA and explain this scheme using a power law [23]. Combining with OBP, we propose parallel OBP (POBP) to solve the big topic modeling problem. We show that POBP has low time, space and communication complexities that it and can converge to the local optimum of the LDA's objective function within the online EM framework [8,17,21].

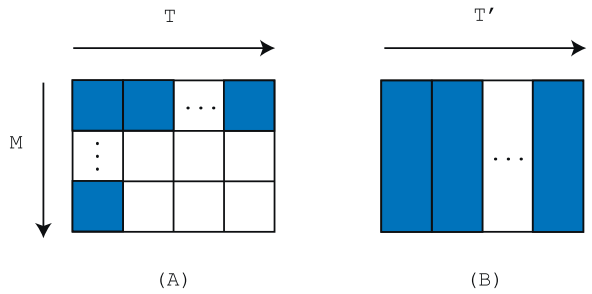

Fig. 1. A comparison of communication costs between (A) parallel online and (B) parallel batch LDA algorithms. Each blue box denotes a communication operation. In (A), the communication rate depends on the number of iterations over all mini-batches MT, while in (B), the communication rate depends on the number of iterations T'. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.1. Communication-Efficient MPA

From (5), there are two straight-forward solutions to reduce the communication cost. The first solution is to reduce the average communication rate T. For example, we may communicate and synchronize the global matrix at every two mini-batch iterations to reduce the communication cost by approximately half. This heuristic solution has been widely used in MPA [22] but suffers from two problems: (1) the lower communication rate may cause a lower topic modeling accuracy, and (2) the overall communication rate also depends on the number of mini-batches M, which is often constrained by each processor's memory space. Therefore, we investigate the second solution to communicate and synchronize only the subset of the global matrix at each mini-batch iteration, i.e., we reduce the size KW in (5). To our best knowledge, there are very few investigations in related work following this research line. We will further explain why selecting the subset of the global matrix dynamically does not significantly influence the topic modeling accuracy based on the power law.

We propose a two-step strategy to select the subset of the global matrix at each iteration in a dynamic manner. First, we select a subset of vocabulary words of size $\lambda_W W$, referred to as the *power words*. For each power word, we select a subset of topics of size $\lambda_K K$, referred to as the *power topics*. In this way, we reduce the communication complexity (5) from KW to $\lambda_K\lambda_W KW$ as follows:

$$\text{Communication complexity} \propto \lambda_K\lambda_W NMTKW, \tag{6}$$

where the ratios $0 < \lambda_K \ll 1$ and $0 < \lambda_W \ll 1$. Obviously, Eq. (6) shows a sublinear complexity of (5). The remaining question concerns how to select both power words and topics.

Our selection criterion is inspired by the residual belief propagation (RBP) [11,45]. At each processor n, we define the residual between message vectors (1) at two successive iterations t and $t-1$:

$$r_{w,d}^{m,n,t}(k) = x_{w,d}^{m,n}|\mu_{w,d}^{m,n,t}(k) - \mu_{w,d}^{m,n,t-1}(k)|, \tag{7}$$

$$r_{w}^{m,n,t}(k) = \sum_{d} r_{w,d}^{m,n,t}(k). \tag{8}$$

We then communicate and synchronize the residual matrix $r_w^{m,n,t}(k)$ across N processors, similar to (4),

$$r_{w}^{m,.,t}(k) = r_{w}^{m,.,t-1}(k) + \sum_{n=1}^{N}[r_{w}^{m,n,t}(k) - r_{w}^{m,.,t-1}(k)]. \tag{9}$$

From (9), we further obtain the synchronized residual vector of vocabulary words:

$$r_{w}^{m,.,t} = \sum_{k} r_{w}^{m,.,t}(k). \tag{10}$$

Finally, we sort vector (10) in descending order and select the power words with the $\lambda_W W$ largest residuals. For each power word, we sort matrix (9) in the Kth dimension and select $\lambda_K K$ power topics for each word with the largest residuals.

Fig. 2 shows an example of the two-step selection method for the global matrix $\hat{\boldsymbol{\phi}}_{4\times 6}$. We set the selection ratios as $\lambda_K = \lambda_W = 0.5$. In the first step, we select three power words with the largest residuals in the vector $r_w^{m,.,t}$, denoted by the blue boxes. In the second step, for each selected power word, we select two power topics with the largest residuals in the matrix $r_w^{m,.,t}(k)$ in the Kth dimension.

This two-step selection process follows the dynamical scheduling scheme. For the mth mini-batch at the first iteration $t = 1$, we need to communicate and synchronize the full matrices $\hat{\boldsymbol{\phi}}_{K\times W}^{m,.,1}$ and $r_{K\times W}^{m,.,1}$. When $2 \le t \le T_m$, we communicate

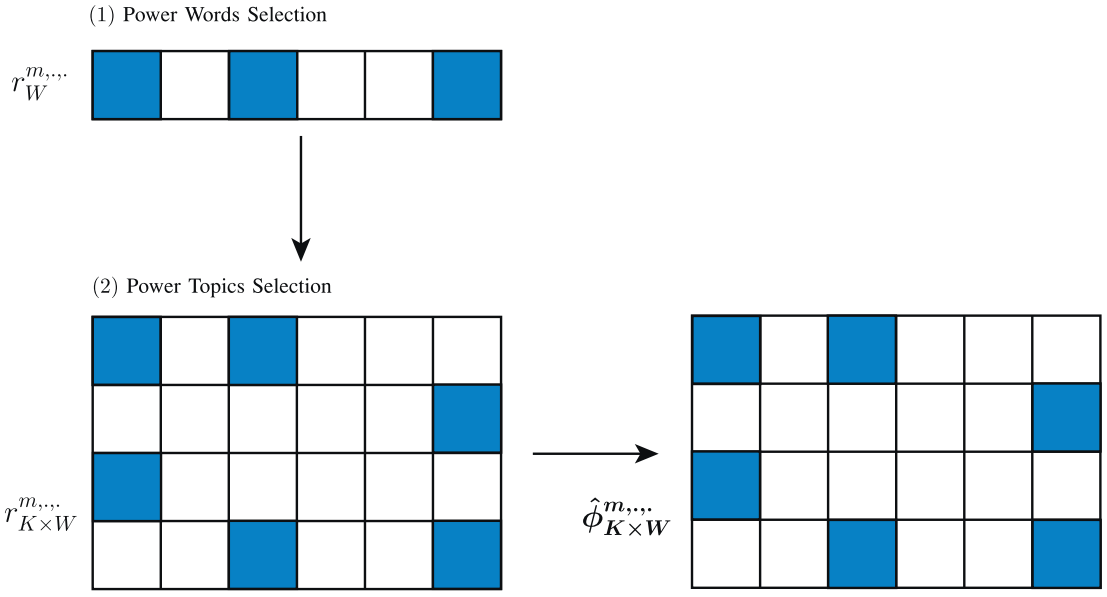

Fig. 2. The two-step power word and topic selection process for a global matrix $\hat{\boldsymbol{\phi}}_{K\times W}$ with $K = 4$ and $W = 6$, where $\lambda_K = \lambda_W = 0.5$. The blue boxes denote the selected power words and topics. In the first step, we select power words by sorting the synchronized residual vector $r_w^{m,t}$. In the second step, for each selected power word, we select further power topics by sorting the synchronized residual matrix $r_w^{m,t}(k)$ in K dimensions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

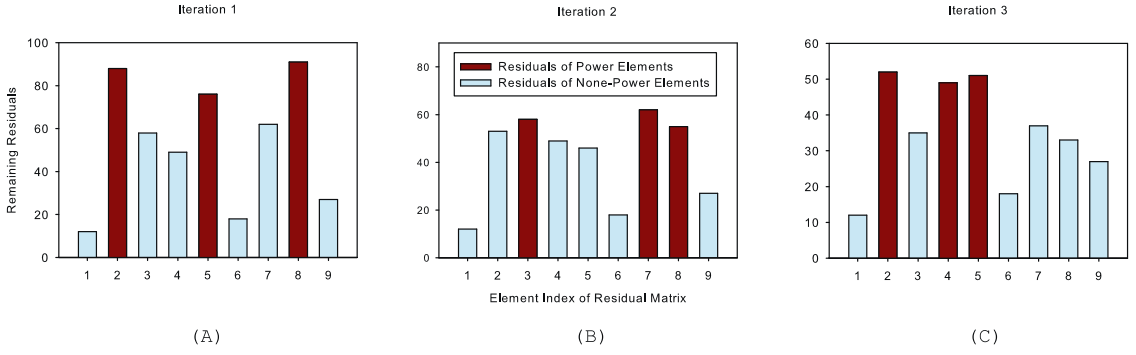

Fig. 3. A dynamic scheduling example of a residual matrix $r_{3\times 3}$ with 3 words and 3 topics, where the 9 elements in $r_{3\times 3}$ are shown in one dimension. (A) In the first iteration $t = 1$, the elements {2, 5, 8} are chosen as power elements. (B) In the second iteration $t = 2$, the elements {3, 7, 8} are chosen as power elements because the residuals of {2, 5} become relatively small. (C) In the third iteration, the elements {2, 4, 5} are selected as the power elements because the residuals of {3, 7, 8} become relatively small.

and synchronize only the partial matrices $\hat{\boldsymbol{\phi}}_{\lambda_K K\times\lambda_W W}^{m,.,2\le t\le T_m}$ and $r_{\lambda_K K\times\lambda_W W}^{m,.,2\le t\le T_m}$, and we keep the remaining elements untouched. The residuals (7) of the power words and topics are becoming increasingly small in the message passing process according to Eq. (1). Therefore, the power words and topics in the previous iteration may no longer be power words or topics due to their relatively small residuals in the next iteration. In this way, all vocabulary words and topics have a chance to be selected as power words or topics before convergence. When all elements in the residual matrix become zero, i.e., $r_w^{m,.,t}(k) \to 0$, the message passing process reaches the convergence state.

For a better understanding of the dynamic scheduling process, Fig. 3 shows an example $r_{3\times 3}^{t=1,2,3}$ at different iterations, where the nine elements are shown in one dimension for simplicity. Fig. 3A shows that, in the first iteration, the elements {2, 5, 8} are selected as the power elements to pass messages such that the residuals for the three elements decrease, while other residuals remain unchanged. Fig. 3B shows that elements {3, 7, 8} are selected as the power elements in the second iteration because the elements {2, 5} produce relatively small residuals. However, they could be power elements again in subsequent iterations when their residuals become large compared to those of other elements. Fig. 3C shows that the elements {2, 4, 5} are chosen as power elements in the third iteration. Therefore, we can guarantee that no information becomes lost since all elements have a chance to become power elements to pass messages, which ensures the topic modeling accuracy of the algorithm.

input : $\mathbf{x}_{W\times D}, K, \lambda_K, \lambda_W, \alpha, \beta$.
output : $\hat{\phi}_{K\times W}$.

1 **for** $m \leftarrow 1$ **to** M **do**
2 **for** *each processor in parallel* $n \leftarrow 1$ **to** N **do**
3 $\mu_{w,d}^{m,n,0}(k) \leftarrow$ random initialization and normalization;
4 $\hat{\theta}_d^{m,n,0}(k) \leftarrow \sum_w x_{w,d}^{m,n}\mu_{w,d}^{m,n,0}(k)$;
5 $\hat{\phi}_w^{m,n,0}(k) \leftarrow \hat{\phi}_w^{m-1,n}(k) + \sum_d x_{w,d}^{m,n}\mu_{w,d}^{m,n,0}(k)$; // stochastic gradient descent
6 $\mu_{w,d}^{m,n,1}(k) \leftarrow$ normalize$([\hat{\theta}_{-w,d}^{m,n,0}(k)+\alpha][\hat{\phi}_{w,-d}^{m,n,0}(k)+\beta]/[\hat{\phi}_{-(w,d)}^{m,n,0}(k)+W\beta])$; // update messages
7 $r_w^{m,n,1}(k) \leftarrow \sum_d x_{w,d}^{m,n}|\mu_{w,d}^{m,n,1}(k) - \mu_{w,d}^{m,n,0}(k)|$; // update residuals
8 $\hat{\theta}_d^{m,n,1}(k), \hat{\phi}_w^{m,n,1}(k) \leftarrow$ update$(\mu_{w,d}^{m,n,1}(k))$; // update sufficient statistics
9 $r_w^{m,.,1}(k) \leftarrow \sum_n r_w^{m,n,1}(k), r_w^{m,.,1} \leftarrow \sum_k r_w^{m,.,1}(k), r_w^{m,n,1}(k) \leftarrow r_w^{m,.,1}$;
10 $\hat{\phi}_w^{m,.,1}(k) \leftarrow \sum_n \hat{\phi}_w^{m,n,1}(k), \hat{\phi}_w^{m,n,1}(k) \leftarrow \hat{\phi}_w^{m,.,1}(k)$;
11 // communicate $r_{K\times W}$ and $\hat{\phi}_{K\times W}$
12 $\lambda_W W \leftarrow$ partial sort$(r_w^{m,.,1}$, 'descend'); // select power words
13 $\lambda_K K \leftarrow$ partial sort$(r_w^{m,.,1}(k)$, 'descend'); // select power topics
14 **for** $t \leftarrow 2$ **to** T **do**
15 **for** $w \in \lambda_W W$ **do**
16 **for** $k \in \lambda_k K$ **do**
17 $\mu_{w,d}^{m,n,t}(k) \leftarrow$ normalize$([\hat{\theta}_{-w,d}^{m,n,t-1}(k)+\alpha][\hat{\phi}_{w,-d}^{m,n,t-1}(k)+\beta]/[\hat{\phi}_{-(w,d)}^{m,n,t-1}(k)+W\beta])$;
18 $r_w^{m,n,t}(k) \leftarrow \sum_d x_{w,d}|\mu_{w,d}^{m,n,t}(k) - \mu_{w,d}^{m,n,t-1}(k)|$;
19 // update the subset of messages and residuals
20 $\hat{\theta}_d^{m,n,t}(k), \hat{\phi}_w^{m,n,t}(k) \leftarrow$ update$(\mu_{w,d}^{m,n,t}(k))$; // update sufficient statistics
21 **end**
22 **end**
23 $r_w^{m,.,t}(k) = r_w^{m,.,t-1}(k) + \sum_n[r_w^{m,n,t}(k) - r_w^{m,.,t-1}(k)], r_w^{m,.,t} \leftarrow \sum_k r_w^{m,.,t}(k), r_w^{m,n,t}(k) \leftarrow r_w^{m,.,t}(k)$;
24 $\hat{\phi}_w^{m,.,t}(k) = \hat{\phi}_w^{m,.,t-1}(k) + \sum_n[\hat{\phi}_w^{m,n,t}(k) - \hat{\phi}_w^{m,.,t-1}(k)], \hat{\phi}_w^{m,n,t}(k) \leftarrow \hat{\phi}_w^{m,.,t}(k)$;
25 // communicate the subsets $r_{\lambda_K K\times\lambda_W W}$ and $\hat{\phi}_{\lambda_K K\times\lambda_W W}$
26 **if** $\sum_w r_w^{m,.,t}/\sum_{w,d} x_{w,d} \le 0.1$ **then** break;
27 $\lambda_W W \leftarrow$ partial sort$(r_w^{m,.,t}$, 'descend'); // select power words dynamically
28 $\lambda_K K \leftarrow$ partial sort$(r_w^{m,.,t}(k)$, 'descend'); // select power topics dynamically
29 **end**
30 **end**
31 **end**

Fig. 4. The POBP algorithm for LDA.

4.2. The POBP algorithm

Although we focus on developing the parallel online belief propagation (POBP) algorithm for big topic modeling tasks, the proposed communication-efficient MPA can be applied to both parallel batch and online LDA algorithms. Fig. 4 summarizes the proposed POBP algorithm. We distribute each incoming mini-batch $x_{w,d}^{m,n}$ to N processors in parallel (line 2). At the first iteration $t = 1$, we randomly initialize and normalize messages $\mu_{w,d}^{m,n,0}$ (line 3), which are used to update the sufficient statistics $\hat{\theta}_d^{m,n,0}(k)$ and $\hat{\phi}_w^{m,n,0}(k)$ using Eqs. (2) and (3) (lines 4 and 5). Note that we use the stochastic gradient descent [6,27] to update (3) in line 5, where the initial $\hat{\phi}^{m=0}$ is set as the zero matrix. Then, we update both messages $\mu_{w,d}^{m,n,1}(k)$ and residuals $r_w^{m,n,1}(k)$ using Eqs. (1) and (7) (lines 6 and 7). The messages are in turn used to update the sufficient statistics $\hat{\theta}_d^{m,n,1}(k)$ and $\hat{\phi}_w^{m,n,1}(k)$ (line 8). At the end of the first iteration, all processors communicate and synchronize two global matrices $\hat{\phi}_w^{m,.,1}(k)$ and $r_w^{m,.,1}(k)$ and transfer the global matrices back to each processor (lines 9 and 10). Using a two-step selection method, we select the power words and topics from the global residual matrix (lines 12 and 13). We use partial sorting to find the power words and topics with the largest $\lambda_W W$ and $\lambda_K K$. The computation cost of the partial sort algorithm is significantly lower than the quick sort algorithm since we do not need complete sorting. In addition, we use the parallel implementations of the partial sort algorithm to further speed-up the selection process. The time complexity of partial sorting is at most $W\log W$ and $K\log K$, where W is the vocabulary size and K is the number of topics.

In the following iterations $2 \le t \le T$, we update only the subsets of messages $\mu_{w,d}^{m,n,t}(k)$ and $r_w^{m,n,t}(k)$ residuals based on the selected power words and topics (lines 17 and 18) and communicate only the subsets of matrices $\hat{\phi}_w^{m,.,t}(k)$ and $r_w^{m,.,t}(k)$ (lines 23 and 24). In the dynamical scheduling process, we select the power words and topics based on the synchronized residual matrix $r_w^{m,.,t}(k)$ (lines 27 and 28). If the average of the residual matrix is below a threshold (line 26), we terminate all processors and load the next mini-batch $x_{w,d}^{m+1,n}$ after freeing the memory, except for the global topic-word matrix $\hat{\phi}_w^{m,\cdots}(k)$. POBP terminates when all M mini-batches have been processed (line 1). When $M \to \infty$, POBP can be viewed as a life-long or never-ending topic modeling algorithm. The output is the global sufficient statistics $\hat{\boldsymbol{\phi}}_{K\times W}$, which can be nor-

Table 2
Comparison of complexities.

Algorithms	Computation cost	Memory cost	Communication cost
POBP	$\eta\lambda_K\lambda_W KWDT/N$	$K(\eta WD+D)/MN+2KW$	$\lambda_K\lambda_W KWMNT$
OBP	$\eta\lambda_K\lambda_W KWDT$	$K(\eta WD+D)/M+2KW$	–
PGS	$\eta' KWDT'/N$	$(K\times D+\eta' WD)/N+KW$	$NKWT'$

malized to obtain the topic-word multinomial parameter matrix $\boldsymbol{\phi}_{K\times W}$. If $N=1$, POBP reduces to the OBP [49] algorithm on a single processor. If $M=1$, POBP reduces to the parallel batch BP algorithm on N processors [38].

4.2.1. Convergence analysis

The objective of LDA is to maximize the joint probability $p(\mathbf{x}, \boldsymbol{\theta}, \boldsymbol{\phi}|\alpha, \beta)$ [3,5,46]. OBP uses the stochastic gradient descent method [6,27] to update the topic-word matrix:

$$\hat{\phi}_w^m(k)=\hat{\phi}_w^{m-1}(k)+\frac{1}{m-1}\Delta\hat{\phi}_w^m(k), \tag{11}$$

where $\Delta\hat{\phi}_w^m(k)=\sum_d x_{w,d}^m\mu_{w,d}^m(k)$ in Eq. (3) is the gradient generated by the current mini-batch. Eq. (11) has a learning rate of $1/(m-1)$ because $\hat{\phi}_w^{m-1}(k)$ accumulates the sufficient statistics of the previous $m-1$ mini-batches, and $\Delta\hat{\phi}_w^m(k)$ accumulates only the sufficient statistics of the current mini-batch. The parameter estimation is invariant to the scaling of the sufficient statistics Eq. (3). Since this learning rate satisfies two conditions,

$$\sum_{m=2}^{\infty}\frac{1}{m-1}=\infty, \tag{12}$$

$$\sum_{m=2}^{\infty}\frac{1}{(m-1)^2}<\infty, \tag{13}$$

the online stochastic approximation [27] proves that the sufficient statistics $\hat{\phi}_w^m(k)$ will converge to a stationary point, and the gradient $\Delta\hat{\phi}_w^m(k)$ will converge to zero when $m\to\infty$. Using (11), OBP can incrementally improve $\hat{\boldsymbol{\phi}}^m$ to maximize the log-likelihood $\ell(\cdot)$ of the joint probability of LDA:

$$\ell(\hat{\boldsymbol{\phi}}^{m+1})\geq\ell(\hat{\boldsymbol{\phi}}^m). \tag{14}$$

In this sense, when $m\to\infty$, OBP can converge to the local optimum of the LDA's log-likelihood function. A more detailed convergence proof of OBP, which is not the focus of this paper, can be found in the published work [49].

Since OBP can converge to the local optimum of LDA's objective function, we show that POBP can also achieve this goal on N processors. For the mth mini-batch, the global $\hat{\phi}_w^{m-1}(k)$ of the previous mini-batch remains unchanged for N processors (line 5). Indeed, all processors update only the local gradient $\Delta\hat{\phi}_w^{m,n,t}(k)$ from the current mini-batch $x_{w,d}^{m,n}$ and communicate this local gradient according to (4) as follows:

$$\Delta\hat{\phi}_w^{m,.,t}(k)=\Delta\hat{\phi}_w^{m,.,t-1}(k)+\sum_{n=1}^{N}[\Delta\hat{\phi}_w^{m,n,t}(k)-\Delta\hat{\phi}_w^{m,.,t-1}(k)], \tag{15}$$

where the synchronized gradient is almost the same as (11). In addition, the learning rate remains $1/(m-1)$, which guarantees the convergence of POBP. If we do not communicate at each iteration, Eq. (15) produces an inaccurate local gradient (11) and thus leads to a low convergence speed. However, from (11) and (15), lowering the communication rate does not change the convergence property of POBP; rather, it reduces its convergence speed. The proposed POBP communicates more frequently than its offline counterparts, as shown in Fig. 1, which ensures its superiority over offline algorithms in terms of convergence performance.

4.2.2. Complexity and scalability

Table 2 compares the complexities of POBP with those of the OBP [49] and PGS [22] algorithms. For simplicity, we assume that the number of non-zero elements in $\mathbf{x}_{W\times D}$ is ηWD, where η is a very small constant value depending on the data sets since $\mathbf{x}_{W\times D}$ is very sparse. Similarly, we assume that the total number of word tokens in $\mathbf{x}_{W\times D}$ is $\eta' WD=\sum_{w,d}x_{w,d}$, where η' is also a constant value depending on the data sets. Generally, $\eta\ll\eta'$ for most data sets. Suppose that the overall computation cost is A, and the communication cost for each processor is B; thus, the overall cost of N processors can be simplified as

$$\text{Overall cost}=\frac{A}{N}+BN, \tag{16}$$

where

$$N^* = \sqrt{\frac{A}{B}} \tag{17}$$

minimizes the overall cost (16) to $2\sqrt{AB}$. From (17), we see that it is the ratio between computation and communication costs that determines the scalability, i.e., the optimal number of processors for the minimum overall cost. Note that, in practice, the communication cost per processor B is a variable that also depends on the bandwidth limitation between processors. When N increases, B will also increase nonlinearly due to the complex communication operations being performed over limited bandwidth. Although Eq. (16) is a simplified estimation of the relationship between computation and communication costs, it provides clues for the estimation of the optimal number of processors in practice. For simplicity, we use (16) and (17) in the following analysis, where we use the size of the communicated and synchronized matrices of each processor in Table 2 to approximate B.

For each mini-batch at the first iteration ($t = 1$), POBP is required to scan the entire mini-batch and communicate two complete matrices $\hat{\boldsymbol{\phi}}_{K\times W}$ and $r_{K\times W}$. In the following iterations, POBP scans only the subset of the mini-batch and communicates the subsets of matrices $\hat{\boldsymbol{\phi}}_{K\times W}$ and $r_{K\times W}$. Since the number of iterations for convergence is often very large (for example, $T \approx 200$), the total computation and communication costs are dominated by the remaining iterations ($2 \le t \le T$). Thus, we approximate the overall computation and communication costs (without considering the small partial sorting costs) shown in Table 2. The real-world costs are proportional to these values. According to (16) and (17), the optimal number of processors is

$$N^* \propto \sqrt{\frac{\eta D}{M}} = \sqrt{\eta D_m}, \tag{18}$$

and the minimal overall cost is

$$\text{POBP's minimum cost} \propto 2\lambda_K\lambda_W KWT\sqrt{\eta DM}. \tag{19}$$

This analysis is consistent with our intuition that the optimal number of processors in POBP scales linearly with the mini-batch size D_m. When $M = 1$, POBP reduces to the parallel batch BP algorithm with the minimum overall cost when the optimal number of processors reaches the maximum.

However, the memory cost of each processor becomes very high, as shown in Table 2, since we have to store the local message matrix $\boldsymbol{\mu}_{K\times \eta WD}$, the document-topic matrix $\hat{\boldsymbol{\theta}}_{K\times D/M}$, the global topic-word matrix $\hat{\boldsymbol{\phi}}_{K\times W}$ and the residual matrix $r_{K\times W}$. Therefore, POBP provides a flexible solution by setting the number of mini-batches M for big topic modeling tasks. When each processor has enough memory space, we can set the smaller number of mini-batches M and use more processors to achieve high speed. When there is no longer sufficient memory for each processor, we can set a larger number of mini-batches M and use fewer processors to obtain a relatively low speed. Note that the minimum overall cost of POBP scales with the square root $\sqrt{DM}$, which is often significantly lower than that of the OBP (e.g., OBP scales linearly with the number of documents D) on a single processor, as shown in Table 2. Moreover, POBP uses less memory for each processor compared to OBP. In this sense, POBP is more suitable than OBP for big topic modeling tasks in real-world applications.

Table 2 also shows the complexities of the PGS algorithm [22], which is one of the widely used big topic modeling solutions introduced in Section 2. Its computation scales linearly with the number of word tokens in $\mathbf{x}_{W\times D}$. According to (16) and (17), the optimal number of processors is $\sqrt{\eta' D}$, and the minimum overall cost is $2KWT'\sqrt{\eta' D}$. Obviously, POBP often obtains a lower minimum cost (19) compared to PGS. Moreover, POBP consumes less memory than PGS, and thus, it is more suitable for big topic modeling tasks. Indeed, if $\lambda_W = 0.1$ and $\lambda_K = 50/K$ (see the experiments in Section 5.1), POBP's minimum cost is insensitive to the number of topics K and the vocabulary size. This is a good property since big data sets often contain a large number of topics and vocabulary words [50]. Although parallel FGS (PFGS) [24] and SGS (PSGS) [39] are also insensitive to the number of topics K, they still consume more memory space than does POBP. In addition, lowering the computation cost instead of the communication cost will degrade the scalability, as shown in Eq. (17), i.e., the optimal number of processors N^* will decrease.

4.3. Power law explanation

The power law, also known as the long-tail principle or the 80/20 rule [28], refers to the fact that a major proportion of effects comes from a small fraction of the causes for many events. We show that the selected power words and topics based on residuals (9) and (10) follow this power law. In this paper, we take the ENRON data set [24] as an example to demonstrate the appearance of the power law. We set the number of topics as 500 and select the messages at the 10th iteration.

First, we show that the residual (7) can be used to evaluate the convergence of the topic modeling process, where the predictive perplexity (10) has been widely used as the convergence condition of LDA algorithms [3,5,46]. Fig. 5 compares the predictive perplexity of LDA and the average residual over all words. We see that the two curves have almost the same trend reflecting the convergence state. This is why we use the average residual as the convergence condition in the POBP algorithm in Fig. 4 (line 26). Intuitively, if the residuals decrease to zero, the message values do not change, and thus, the

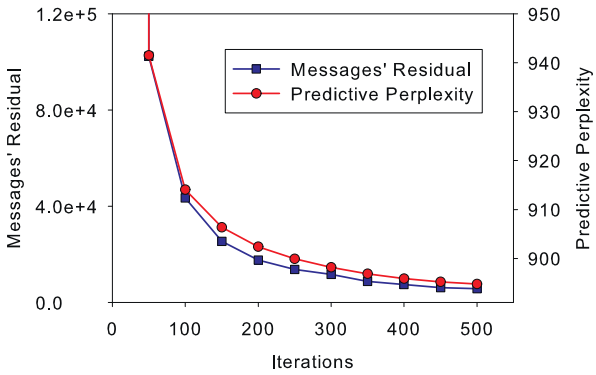

Fig. 5. The residual (blue curve) and predictive perplexity (red curve) as a function of the number of iterations on ENRON. The predictive perplexity decreases with the residual, which indicates the convergence. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

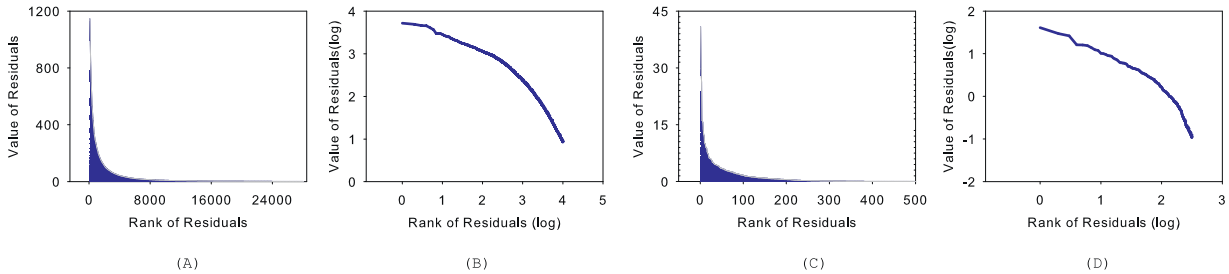

Fig. 6. The message value as a function of the message rank when $K = 500$ at the 10th iteration on ENRON. (A) Linear plot of the message rank of the vocabulary words. (B) Log-log plot of the message rank of the vocabulary words. (C) Linear plot of the message rank of the topics. (D) Log-log plot of the message rank of the topics.

parameters are fixed at the local optimum. In this sense, we can speed-up convergence by minimizing the larger residuals first and then the smaller residuals. This is the first motivation of our two-step selection method in communication-efficient MPA in Section 4.1.

Second, we show that the distribution of residuals approximately follows the power law at each mini-batch iteration. A simple way to identify power-law behavior in either natural or man-made systems is to draw a histogram with both axes plotted on logarithmic scales, called a log-log plot [28]. If the log-log plot approximates a straight line, we consider that a power law applies. We sort the residuals in descending order. We draw the linear plot for r_w in Fig. 2, with the x-axis indicating the residual ranks and the y-axis indicating the residual values. Fig. 6A indicates that a small fraction of words represent a very large proportion of the residuals. Fig. 6B shows that the corresponding log-log plot approximately follows a power law. This phenomenon confirms that only a small subset of vocabulary words contributes almost all residual values. More specifically, the top 10% of words account for 79% of the total residual values, and the top 20% of words account for almost 90% of the total residual values. Therefore, it is efficient to minimize the residuals of those power words first to speed-up the convergence. Fig. 6C shows the linear plot for $r_w(k)$ in Fig. 2, and Fig. 6D shows the corresponding log-log plot. Both figures confirm that the residual distribution of power topics approximately follows a power law. Therefore, we only need to perform computation and communication for power words and topics, which will be updated through the dynamical scheduling at each iteration in Fig. 3.

5. Experiments

We compare the proposed POBP with parallel FGS (PFGS) [24], parallel SGS (PSGS) [39], Yahoo LDA (YLDA) [1], and parallel variational Bayes (PVB) [51]. All these benchmark algorithms have open-source codes. To ensure a fair comparison, we re-write their source codes in the C++ language [44]. In addition, we use the integer type to store the LDA parameters in the GS-based algorithms, whereas we use the single-precision floating-point format to represent the LDA parameters in both the PVB and POBP algorithms. Such an implementation difference is caused by the sampling process in the GS-based algorithms [13].

We run the above algorithms on a cluster with up to 1024 processors (1.9 GHz CPU, 2 GB of memory) to perform the experiments. All the processors communicate through a high-speed Infiniband with a bandwidth of 20 GB per second. The

Table 3
Summary of the data sets.

Data sets	D	W	N_{token}	NNZ	Size (M)
NIPS	1500	5821	1,829,267	861,513	8.11
ENRON	39,861	6536	6,412,172	2,374,385	28.34
NYTIMES	300,000	7871	99,542,125	44,379,275	568.88
WIKIPEDIA	4,360,095	5363	665,375,061	154,934,308	1983.77
PUBMED	8,200,000	6902	737,869,083	222,399,377	3043.04

cluster runs MPICH2[1], one of the most popular implementations of the message-passing standard for distributed-memory applications. Following [24], we use the fixed hyper-parameters $\alpha = 2/K$ and $\beta = 0.01$ for all algorithms to guarantee a fair comparison. To reach the convergence state, we run PFGS, PSGS, YLDA and PVB using 500 iterations [22]. For POBP, we set $NNZ \approx 45,000$ in each mini-batch since OBP's performance is insensitive to the mini-batch size [49]. In addition, this mini-batch size can be easily fit into the 2 GB of memory of each processor. We evenly distribute D documents to N processors to avoid any load imbalance.

We use five publicly available data sets: NIPS, ENRON, NYTIMES, PUBMED [24] and WIKIPEDIA.[2] The NIPS data set is quite small and contains only 1500 documents. The ENRON data set is a relatively middle-size data set, and we use it for parameter tuning. The other three data sets are relatively large, with up to 8 million documents; we use them for the web-scale experiments. We follow [14] and remove the words from a fixed truncated vocabulary to obtain a shorter vocabulary since some vocabulary words occur rarely and contribute minimally to topic modeling. Although the vocabulary size W has been greatly reduced, most of the word tokens N_{token} and non-zero-elements NNZ are still reserved. For example, although we reduce the vocabulary size of PUBMED from 141,043 to 6902, corresponding to a ratio of 4.89%, we reduce the number of word tokens from approximately 7 million to 3 million, corresponding to a ratio of over 40%. As a result, we can fit the word-topic distribution $\boldsymbol{\phi}_{K\times W}$ in the 2 GB of memory of each processor when K is large. Table 3 summarizes the statistics of the data sets, where D denotes the number of documents, W is the vocabulary size, N_{token} is the number of word tokens, NNZ is the number of non-zero elements, and "Size (M)" is the size of the data sets in MBytes.

We use the predictive perplexity ($\mathcal{P}$) [3,46] to measure the accuracy of different parallel LDA algorithms. To calculate the predictive perplexity, we randomly partition each document into subsets of 80% and 20%. Fixing the word-topic distribution $\boldsymbol{\phi}_{K\times W}$, we estimate $\boldsymbol{\theta}_{K\times D}$ on the 80% subset by the training algorithms from the same random initialization after 500 iterations. Then, we calculate the predictive perplexity on the remaining 20% subset:

$$\mathcal{P} = \exp\left\{-\frac{\sum_{w,d} x^{20\%}_{w,d}\log\left[\sum_k \theta_d(k)\phi_w(k)\right]}{\sum_{w,d} x^{20\%}_{w,d}}\right\}, \tag{20}$$

where $x^{20\%}_{w,d}$ denotes the word counts in the 20% subset. A lower predictive perplexity represents a higher accuracy.

5.1. Ratios λ_W and λ_K

POBP introduces two parameters, λ_W and λ_K, to control the ratio of power words and topics at each iteration. The parameter λ_K determines the ratio of power topics evolved at each iteration. A smaller λ_K will lead to reduced computation and communication costs However, this may also result in a lower topic modeling accuracy. In practice, each word may not be allocated to many topics, and thus, $\lambda_K K$ is often a fixed value. To study the effect of different $\lambda_K K$, we evaluate a range of $\lambda_K K$ values on the ENRON data set when $K = 500$.

Fig. 7A shows the predictive perplexity and training time as a function of λ_W when fixing $\lambda_K = 1$, where $\lambda_W = 1$ denotes that all the vocabulary words are scanned at each iteration. We decrease the value of λ_W from 0.4 to 0.025 in an exponential manner. As the training time decreases with decreasing λ_W, the predictive perplexity also increases, indicating a degraded performance. However, when $\lambda_W \geq 0.1$, the increase in perplexity is so small that it can be neglected. This result confirms that a subset of power words at each iteration contributes almost all the topic modeling performance. In addition, we see that a small value of λ_W may lead to an increase in perplexity. For example, when $\lambda_W = 0.025$, the predictive perplexity increases by approximately 8% to 526.8.

Fig. 7B shows the predictive perplexity and training time as functions of $\lambda_K K$ when fixing $\lambda_W = 1$, where $\lambda_K K = 500$ means that all the topics are scanned at each iteration. We change $\lambda_K K$ from 30 to 70 in steps of 10. The results show that the predictive perplexity increases slightly and that the training time decreases steadily with decreasing $\lambda_K K$. Fig. 7B also confirms that a subset of power topics plays an important role in topic modeling. Finally, we combine different values of λ_W and $\lambda_K K$. Fig. 7C shows that $\{\lambda_W = 0.1, \lambda_K K = 50\}$ can achieve a reasonable speedup while ensuring a high accuracy (e.g., the predictive perplexity change is within 15). We also use this setting in Subsection 4.2.2 for the complexity and scalability analysis.

[1] http://www.mpich.org/ .
[2] http://en.wikipedia.org .

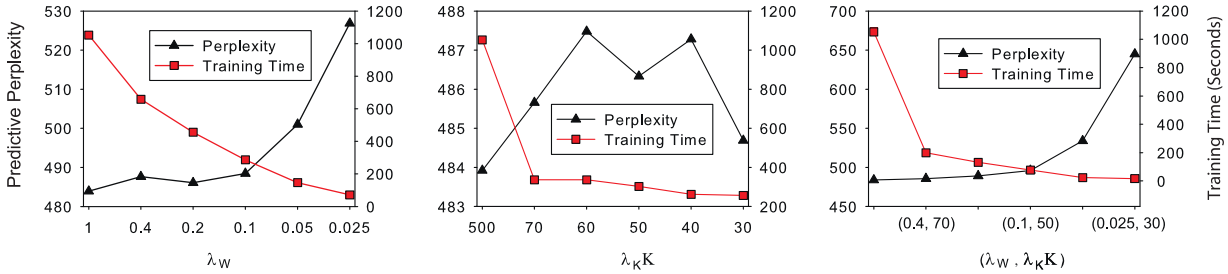

Fig. 7. The predictive perplexity and training time as a function of λ_K and λ_W on the ENRON data set. We fix $K = 500$ and use 12 processors. The left axis denotes the predictive perplexity, and the right axis denotes the training time in seconds. (A) Fixing $\lambda_K = 1$, we test different $\lambda_W = \{0.025, 0.05, 0.1, 0.2, 0.4, 1\}$. (B) Fixing $\lambda_W = 1$, we test different $\lambda_K K = \{30, 40, 50, 60, 70, 500\}$. (C) We test some combinations of λ_W and $\lambda_K K$. We see that, when $\lambda_W = 0.1$ and $\lambda_K K = 50$, POBP can achieve a significant speedup while obtaining a good accuracy.

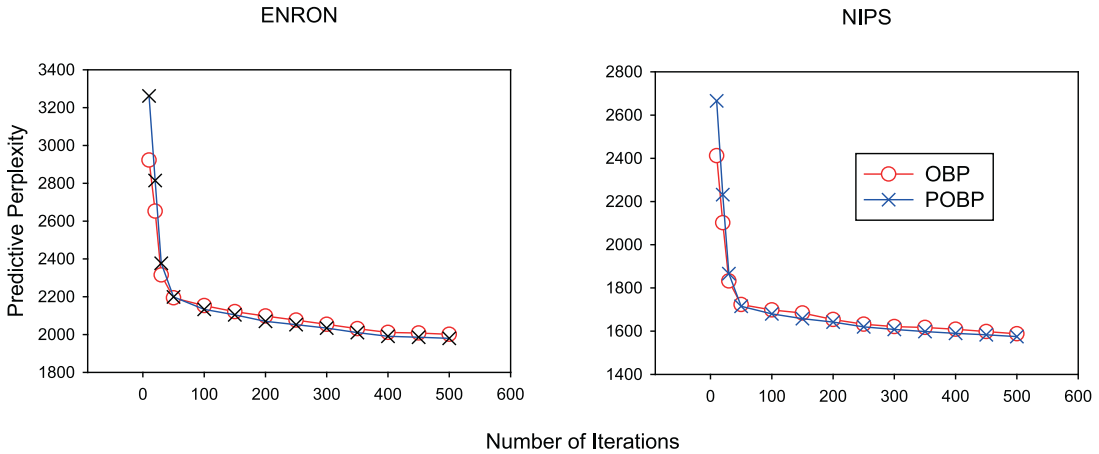

Fig. 8. The predictive perplexity as a function of the iterations on ENRON and NIPS when $K = 500$ using 128 processors. We see that POBP converge slower than OBP within 30 iterations. However, POBP converges as fast as OBP in subsequent iterations and finally achieves a predictive perplexity that is slightly lower than that of OBP.

POBP introduces two parameters, λ_W and λ_K, to control the ratios of power words and topics at each iteration. We set λ_W to 0.025 in the experiments to achieve a balance between accuracy and speed. However, we observe that the ideal value of the parameter λ_K is related to the number of topics K. Each word may not be associated with all topics due to sparsity, and a fixed number of topics contribute almost all the topic modeling performance, as shown in Fig. 7C. As a result, we fix $\lambda_K K$ to 50, which is sufficient to achieve an acceptable accuracy in practice. The larger number of topics K results in the smaller $\lambda_K = 50/K$. Therefore, the communication complexity is relatively independent of the number of topics K when K is very large, which is a sublinear complexity of (5).

5.2. Accuracy

We show that POBP can achieve the same convergence performance as its sequential counterpart, OBP. Moreover, we compare POBP with several state-of-the-art parallel LDA algorithms and show that POBP has a good accuracy property.

The proposed POBP is a distributed version of OBP. Fig. 8 shows the predictive perplexity versus the number of training iterations on the NIPS and ENRON data sets with $K = 500$. The overlapped convergence curves of POBP and OBP show that both POBP and OBP achieve a similar convergence performance. POBP converges slower than OBP during the initial iterations. This is due to the low ratios of power words and topics at each iteration, where the ratios are controlled by two parameters, λ_W and λ_K. However, POBP converges as fast as OBP after 30 iterations and achieves a slightly lower predictive perplexity than does OBP. Since POBP usually requires several hundred iterations for convergence, the lower convergence speed during the earlier stages does not affect the overall convergence performance of POBP in practice. Therefore, POBP can achieve almost the same convergence performance as OBP at each iteration.

Fig. 9 shows the predictive perplexity as a function of training time (in second log-scale) on NYTIMES, PUBMED and WIKIPEDIA using 256 processors when $K = 2000$. We see that POBP converges fastest among all the algorithms, around 10 to 100 times faster than GS-based algorithms and 50 to 400 times faster than PVB. This result is consistent with our convergence analysis in Section 4.2.1. In addition, POBP always achieves the lowest predictive perplexity, indicating its good convergence property. Fig. 10 also shows that POBP yields the lowest predictive perplexity on all data sets given different

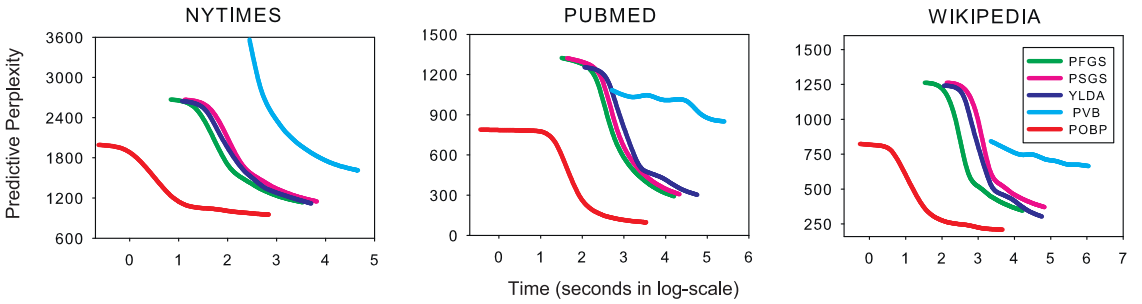

Fig. 9. Predictive perplexity as a function of training time (in second log-scale) on the NYTIMES, PUBMED and WIKIPEDIA data sets using 256 processors when $K = 2000$.

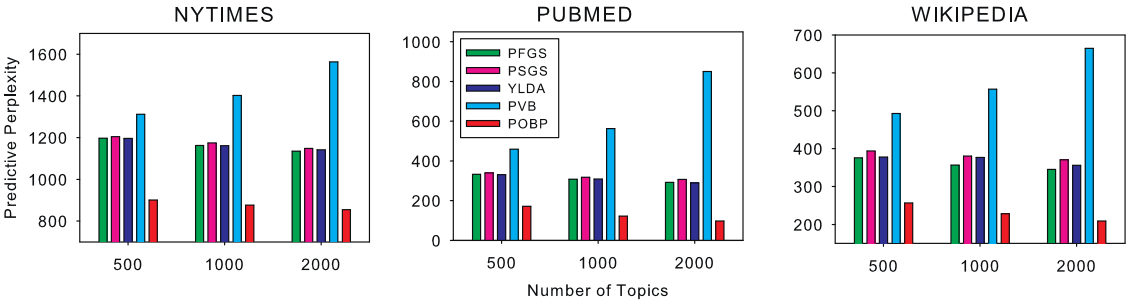

Fig. 10. Comparison of predictive perplexity for all algorithms on the NYTIMES, PUBMED and WIKIPEDIA data sets using 256 processors, where the number of topics $K \in \{500, 1000, 2000\}$.

Table 4
Perplexity gap between POBP and PFGS.

K	NYTIMES	WIKIPEDIA	PUBMED
500	24.41%	31.64%	48.54%
1000	24.57%	36.07%	60.46%
2000	24.69%	39.51%	66.68%

numbers of topics on 256 processors. GS-based algorithms, such as PFGS, PSGS and YLDA, obtain a slightly higher perplexity, and PVB produces the highest perplexity. These results are consistent with observations in previous work [3,46,48]. We see that the predictive perplexity of PVB increases with the number of topics partly due to the overfitting phenomenon.

Table 4 compares the perplexity gap between POBP and PFGS calculated by

$$\text{gap} = \frac{\mathcal{P}_{\text{PFGS}} - \mathcal{P}_{\text{POBP}}}{\mathcal{P}_{\text{PFGS}}} \times 100\%, \tag{21}$$

where $\mathcal{P}$ is the predictive perplexity (20). When $K = 500$, the gap is approximately 24.41% on the relatively small data set NYTIMES, but the gap increases to 31.64% and 48.54% on the larger data sets WIKIPEDIA and PUBMED, respectively. Moreover, the gap increases for all data sets when K increases from 500 to 2000. Such an excellent predictive performance makes POBP a very competitive topic modeling algorithm on real-world big data streams.

5.3. Communication time

Fig. 11 shows the communication time (in second log-scale) of all algorithms on the NYTIMES, PUBMED and WIKIPEDIA data sets using 256 processors when $K \in \{500, 1000, 2000\}$. We see that POBP consumes approximately 5% $\sim$ 20% of the communication time of other algorithms on all data sets. Among all the algorithms, PVB has the longest communication time because the topic-word distribution $\hat{\boldsymbol{\phi}}_{K\times W}$ in PVB is of the single-precision floating type, leading to an approximately doubled communication cost compared to that of GS-based algorithms using the integer type. Although POBP also stores $\hat{\boldsymbol{\phi}}_{K\times W}$ in the single-precision floating-point format, it selects only a subset of the matrix $\hat{\boldsymbol{\phi}}_{K\times W}$ for communication in Subsection 4.1. Hence, POBP is more communication efficient than GS-based algorithms. According to the analysis in Subsection 4.2.2, the total communication time of POBP is proportional to the number of mini-batches M. In our experiments, the number of mini-batches on NYTIMES, PUBMED and WIKIPEDIA is 6, 19 and 17, respectively. Therefore, POBP obtains the smallest total communication time on NYTIMES. This result suggests that, if the memory is sufficient, we should attempt to minimize the number of mini-batches M in POBP to achieve the minimum communication time.

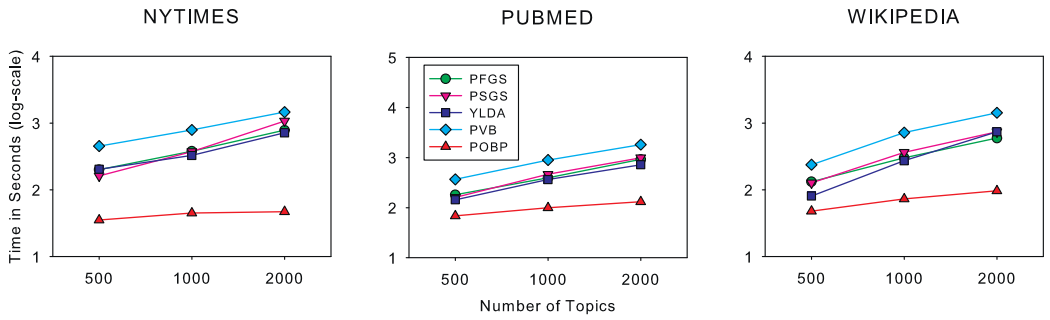

Fig. 11. The communication time (in second log-scale) on NYTIMES, PUBMED and WIKIPEDIA using 256 processors when $K \in \{500, 1000, 2000\}$.

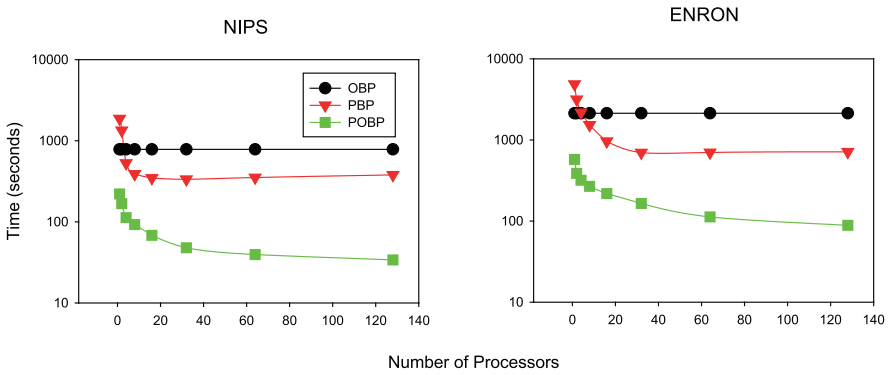

Fig. 12. Training time in seconds (log-scale) of POBP, OBP and PBP on the NIPS and ENRON data sets when $K = 500$ using 1, 2, 4, ..., 128 processors.

5.4. Speed and scalability

We compare POBP with OBP and PBP (parallel belief propagation, the direct combination of BP and MPA following [22]) on both the NIPS and ENRON datasets in terms of speedup performance. Fig. 12 shows the training time in seconds (log-scale) of POBP, OBP and PBP on the NIPS and ENRON data sets when $K = 500$ using 1, 2, 4, ... , 128 processors. The experimental results show that the speedup performance of POBP is much better than those of both OBP and PBP. For example, POBP is approximately 20 times faster than OBP and 10 times faster than PBP on the NIPS data set with 128 processors. The results confirm that the power words and topics can effectively reduce the running time of the algorithms. In addition, POBP demonstrates much better scalability performance than its off-line counterpart PBP because of the lower communication cost. Fig. 12 shows that the training time of POBP continues decreasing with increasing number of processors; however, the training time of PBP remains almost unchanged and even increases when the number of processors reaches a certain threshold. For example, the training time increases when the number of processors exceeds 32 on the NIPS data set. The underlying reason for this phenomenon lies in the computation and communication costs when increasing the number of processors. The computation cost continues decreasing with increasing number of processors because fewer documents are allocated to each processor. However, the communication cost increases because the processors must wait for the slowest processor in the synchronization step [22,33]. Therefore, the total running time of parallel LDA algorithms may increase with increasing number of processors because the communication cost will exceed the computation cost to be the main cost, as illustrated in Fig. 12. We see that POBP has a better scalability performance than PBP because POBP is communication efficient and because its communication cost is only a small fraction of that in PBP.

We find that parallel LDA algorithms perform better on large data sets than on small data sets. On large data sets, the computation cost is usually the main cost because a large number of documents are allocated to each processor, leading to a relatively low communication cost. However, on small data sets, the communication cost becomes the primary cost when the number of processors becomes large. Thus, the overall running time increases, leading to a poor scalability. Moreover, this paper mainly focuses on big topic modeling problems. To this end, we use larger data sets, such as NYTIMES, PUBMED and WIKIPEDIA, for the remaining experiments in this paper.

Fig. 13 shows the training time of all algorithms as a function of the number of topics. We see that POBP is the fastest among all algorithms. PFGS, PSGS and YLDA have comparable speeds, and PVB is the slowest. On all data sets, POBP is approximately 5 to 100 times faster than other algorithms. Such a high speed has been largely attributed to three causes. First, POBP has the smallest communication time, as shown in Fig. 11. Second, POBP runs fast at each iteration because it selects a subset of words and topics for computation, as shown in Fig. 4. Finally, POBP converges very fast, as shown in Fig. 9.

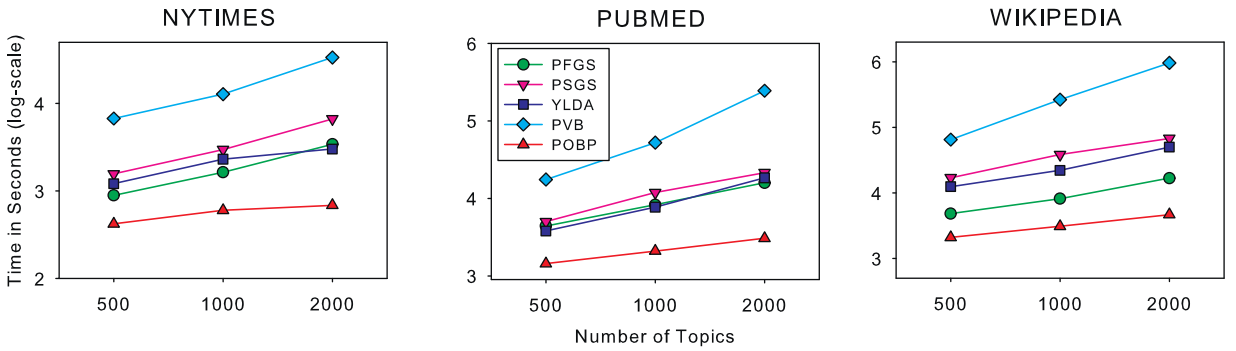

Fig. 13. Training time in seconds (log-scale) of all algorithms on the NYTIMES, PUBMED and WIKIPEDIA data sets when $K \in \{500, 1000, 2000\}$ using 256 processors.

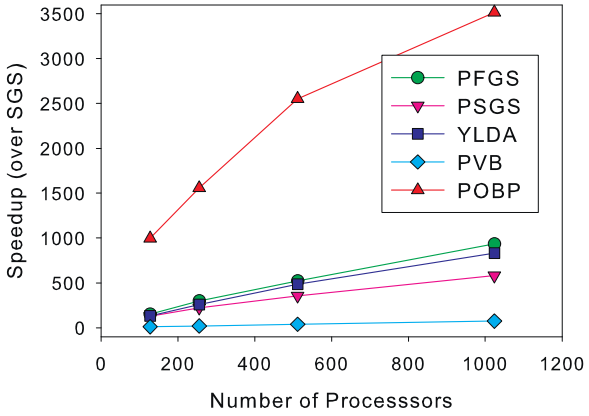

Fig. 14. The speedup performance when $K = 2000$. We choose 1/128 of the training time of PSGS [39] on 128 processors as the baseline.

Table 5
Memory usage (MB) on PUBMED when $K = 2000$.

N	PFGS	PSGS/YLDA	PVB	POBP
1024	349	279	438	1,133
512	541	349	560	1,133
256	924	487	804	1,133
128	1,690	765	1,293	1,133
64	*N/A*	1320	*N/A*	1,133
32	*N/A*	*N/A*	*N/A*	1,133

We use the speedup performance with the number of processors [22] to evaluate the scalability of parallel algorithms. We choose the 1/128 training time of PSGS on 128 processors as the baseline, which approximates the training time of SGS on a single processor without parallelization. Then, the speedup is calculated as the ratio between the baseline and the training time of other parallel algorithms. Fig. 14 shows the speedup performance of all algorithms on PUBMED when $K = 2000$. We show the speedup curve on $N \in \{128, 256, 512, 1024\}$ processors. Although the speedup curve of POBP bends earlier than do those of other algorithms, POBP always achieves a much better speedup performance than other parallel algorithms. This phenomenon confirms that POBP requires only a small number of processors N^* in (18) to achieve the best speedup performance, whereas other parallel algorithms often need more processors to achieve this performance. Moreover, the best performance of POBP is much better than those of other algorithms following the analysis in Section 4.2.2. In this sense, POBP is characterized by good scalability because it uses the fewest number of processors to achieve a much better speedup performance compared to other parallel algorithms.

5.5. Memory usage

Big topic modeling tasks are often limited by the memory space of each processor. Table 5 shows the memory usage of all algorithms on each processor on the PUBMED data set when $K = 2000$. The memory usage of PSGS, YLDA, PFGS and PVB decreases with increasing number of processors, whereas POBP consumes a constant memory space. The major reason

for this is that parallel batch LDA algorithms can distribute both the data $\mathbf{x}_{W\times D}$ and document-topic matrix $\hat{\boldsymbol{\theta}}_{K\times D}$ to N processors, and thus, the total memory usage of each processor will decrease linearly with N. However, when N is small, parallel batch LDA algorithms may not load $1/N$ data and the document-topic matrix into memory for computation (e.g., when $N \leq 64$, PFGS and PVB fail to process PUBMED in Table 5). On the other hand, POBP is an online algorithm that loads only a mini-batch of the data and document-topic matrix into memory, representing a constant value dependent on the mini-batch size D_m. In practice, users can provide D_m according to each processor's memory quota. Generally, we maximize D_m to reduce M to achieve the minimum communication time (19). To further reduce the memory usage of POBP, we may use a hard disk as extended memory to store the word-topic matrix $\hat{\boldsymbol{\phi}}_{K\times W}$, as in [49]. Another strategy is to distribute $\hat{\boldsymbol{\phi}}_{K\times W}$ to N processors by increasing the communication costs. In this way, we can extract more topics from more vocabulary words without truncation in our experimental settings.

6. Conclusions

This paper proposes a novel parallel multi-processor architecture (MPA) for big topic modeling tasks. This communication-efficient MPA can be combined with both batch and online LDA algorithms. For example, we combine this MPA with OBP [49], referred to as the POBP algorithm, for big data streams in this paper. At each iteration, POBP computes and communicates subsets of vocabulary words and topics, called power words and topics, and thus has very low computation, memory and communication costs. Extensive experiments on big data sets confirm that POBP is faster, lighter, and more accurate than other state-of-the-art parallel LDA algorithms such as parallel fast Gibbs sampling (PFGS) [24], parallel sparse Gibbs sampling (PSGS) [39], Yahoo LDA (YLDA) [1], and parallel variational Bayes (PVB) [51]. Therefore, POBP is very competitive for web-scale topic modeling applications, which require a high processing speed under limited resources or seek a high processing efficiency/cost performance. Since POBP can be interpreted within the EM framework, its basic idea can be generalized to speed-up parallel batch or online EM algorithms for other latent variable models. In addition, the power law explanation may shed additional light on building faster big data learning algorithms such as deep learning algorithms with high performance computing systems [9].

Future work may include two parts. First, we still need to investigate multi-core architectures (MCAs) such as GPU clusters for big topic modeling in shared memory systems [52]. We may be able to avoid serious race conditions by dynamical scheduling non-conflicting subsets of vocabulary words and topics. Second, we need to study how to apply POBP to other parallel paradigms such as in-memory Map-Reduce (Spark)[3] or Graph-Lab/Chi.[4]

Acknowledgments

This work is supported by NSFC (grant nos. 61272449, 61373092 and 61572339), the Science and Technology Support Program of Jiangsu (grant no. BE2014005), and a GRF grant from RGC UGC Hong Kong (GRF Project No. 9041574).

References

[1] A. Ahmed, M. Aly, J. Gonzalez, S. Narayanamurthy, A. Smola, Scalable inference in latent variable models, in: WSDM, 2012, pp. 123–132.
[2] S. Arora, R. Ge, Y. Halpern, D. Mimno, A. Moitra, D. Sontag, Y. Wu, M. Zhu, A practical algorithm for topic modeling with provable guarantees, ICML, 2013.
[3] A. Asuncion, M. Welling, P. Smyth, Y.W. Teh, On smoothing and inference for topic models, in: UAI, 2009, pp. 27–34.
[4] D.M. Blei, Introduction to probabilistic topic models, Commun. ACM (2012) 77–84.
[5] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent Dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.
[6] L. Bottou, Online Learning and Stochastic Approximations, Cambridge University Press, 1998.
[7] L. Cao, F.F. Li, Spatially coherent latent topic model for concurrent segmentation and classification of objects and scenes, Proceedings (2007) 1–8.
[8] O. Cappé, E. Moulines, Online expectation-maximization algorithm for latent data models, J. R. Stat. Soc. Ser. B 71 (3) (2009) 593–613.
[9] A. Coates, B. Huval, T. Wang, D.J. Wu, A.Y. Ng, B. Catanzaro, Deep learning with COTS HPC systems, ICML, 2013.
[10] M. Crasso, C. Mateos, A. Zunino, M. Campo, Easysoc: making web service outsourcing easier, Inf. Sci. 259 (2014) 452–473.
[11] G. Elidan, I. McGraw, D. Koller, Residual belief propagation: informed scheduling for asynchronous message passing, in: UAI, 2006, pp. 165–173.
[12] Y. Gao, Z. Sun, Y. Wang, X. Liu, J. Yan, J. Zeng, A comparative study on parallel LDA algorithms in MapReduce framework, 2015.
[13] T.L. Griffiths, M. Steyvers, Finding scientific topics, Proc. Natl. Acad. Sci. 101 (2004) 5228–5235.
[14] M. Hoffman, D. Blei, F. Bach, Online learning for latent Dirichlet allocation, in: NIPS, 2010, pp. 856–864.
[15] S. Kinoshita, T. Ogawa, M. Haseyama, Lda-based music recommendation with cf-based similar user selection, in: IEEE Global Conference on Consumer Electronics, 2015.
[16] K. Lagus, S. Kaski, T. Kohonen, Mining massive document collections by the WEBSOM method, Inf. Sci. 163 (1¿C3) (2004) 135–156. Soft Computing Data Mining
[17] P. Liang, D. Klein, Online EM for unsupervised models, in: Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the ACL, 2009, pp. 611–619.
[18] X. Liu, J. Zeng, X. Yang, J. Yan, Q. Yang, Scalable parallel em algorithms for latent Dirichlet allocation in multi-core systems, in: Proceedings of the 24th International Conference on World Wide Web, ACM, 2015, pp. 669–679.
[19] Z. Liu, Y. Zhang, E.Y. Chang, M. Sun, Plda+: parallel latent Dirichlet allocation with data placement and pipeline processing, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 1–18.
[20] D. Mimno, M.D. Hoffman, D.M. Blei, Sparse stochastic inference for latent Dirichlet allocation, ICML, 2012.
[21] R.M. Neal, G.E. Hinton, A view of the EM algorithm that justifies incremental, sparse, and other variants, vol. 89 (1998) 355–368.

[3] http://spark.incubator.apache.org/ .
[4] http://graphlab.org/

[22] D. Newman, A. Asuncion, P. Smyth, M. Welling, Distributed algorithms for topic models, J. Mach. Learn. Res. 10 (2009) 1801–1828.
[23] M. Newman, Power laws, Pareto distributions and Zipf's law, Contemp. Phys. 46 (5) (2005) 323–351.
[24] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, M. Welling, Fast collapsed Gibbs sampling for latent Dirichlet allocation, in: KDD, 2008, pp. 569–577.
[25] Y. Rao, Q. Li, X. Mao, L. Wenyin, Sentiment topic models for social emotion mining, Inf. Sci. 266 (2014) 90–100.
[26] F. Ren, M.G. Sohrab, Class-indexing-based term weighting for automatic text classification, Inf. Sci. 236 (2013) 109–125.
[27] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Stat. 22 (3) (1951) 400–407.
[28] R. Sanders, The Pareto principle: its use and abuse, J. Product Brand Manage. 1 (2) (1992) 37–40.
[29] A. Smola, S. Narayanamurthy, An architecture for parallel topic models, in: PVLDB, 2010, pp. 703–710.
[30] J. Stefanowski, A. Cuzzocrea, D. Ślezak, Processing and mining complex data streams, Inf. Sci. 285 (2014) 63–65. Processing and Mining Complex Data Streams
[31] M. Steyvers, T. Griffiths, Probabilistic topic models, Handb. Lat. Seman. Anal. 427 (7) (2007) 424–440.
[32] M. Wahabzada, K. Kersting, Larger residuals, less work: active document scheduling for latent Dirichlet allocation, in: ECML/PKDD, 2011, pp. 475–490.
[33] Y. Wang, H. Bai, M. Stanton, W.Y. Chen, E. Chang, PLDA: parallel latent Dirichlet allocation for large-scale applications, in: Algorithmic Aspects in Information and Management, 2009, pp. 301–314.
[34] Y. Wang, X. Zhao, Z. Sun, H. Yan, L. Wang, Z. Jin, L. Wang, Y. Gao, C. Law, J. Zeng, Peacock: learning long-tail topic features for industrial applications, ACM Trans. Intell. Syst. Technol. (TIST) 6 (4) (2015) 47.
[35] L. Wenyin, X. Quan, M. Feng, B. Qiu, A short text modeling method combining semantic and statistical information, Inf. Sci. 180 (20) (2010) 4031–4041.
[36] M.-S. Wu, Modeling query-document dependencies with topic language models for information retrieval, Inf. Sci. 312 (2015) 1–12.
[37] F. Yan, N. Xu, Y. Qi, Parallel inference for latent Dirichlet allocation on graphics processing units, in: NIPS, 2009, pp. 2134–2142.
[38] J. Yan, Z.-Q. Liu, Y. Gao, J. Zeng, Communication-efficient parallel belief propagation for latent Dirichlet allocation, arXiv:1206.2190v1[cs.LG], (2012).
[39] L. Yao, D. Mimno, A. McCallum, Efficient methods for topic model inference on streaming document collections, in: KDD, 2009a, pp. 937–946.
[40] L. Yao, D. Mimno, A. McCallum, Efficient methods for topic model inference on streaming document collections, 2009b.
[41] J. Yin, Q. Ho, E.P. Xing, A scalable approach to probabilistic latent space inference of large-scale networks., Adv. Neural Inf. Process. Syst. 2013 (2013) 422–430.
[42] H.F. Yu, C.J. Hsieh, H. Yun, S.V.N. Vishwanathan, I.S. Dhillon, A scalable asynchronous distributed algorithm for topic modeling, Comput. Sci. (2014).
[43] J. Yuan, F. Gao, Q. Ho, W. Dai, J. Wei, X. Zheng, E.P. Xing, T.Y. Liu, W.Y. Ma, Lightlda: big topic models on modest compute clusters, Comput. Sci. (2014).
[44] J. Zeng, A topic modeling toolbox using belief propagation, J. Mach. Learn. Res. 13 (2012) 2233–2236.
[45] J. Zeng, X.-Q. Cao, Z.-Q. Liu, Residual belief propagation for topic modeling, in: ADMA, 2012a, pp. 739–752.
[46] J. Zeng, W.K. Cheung, J. Liu, Learning topic models by belief propagation, IEEE Trans. Pattern Anal. Mach. Intell. 35 (5) (2013) 1121–1134.
[47] J. Zeng, Z.-Q. Liu, X.-Q. Cao, A new approach to speeding up topic modeling, arXiv:1204.0170(2012b).
[48] J. Zeng, Z.-Q. Liu, X.-Q. Cao, A new approach to speeding up topic modeling, arXiv:1204.0170 [cs.LG]. (2012c)
[49] J. Zeng, Z.-Q. Liu, X.-Q. Cao, Fast online EM for big topic modeling, IEEE Trans. Knowledge Data Eng. (2016).
[50] K. Zhai, J. Boyd-Graber, Online latent Dirichlet allocation with infinite vocabulary, in: ICML, 2013, pp. 561–569.
[51] K. Zhai, J. Boyd-Graber, N. Asadi, Mr. LDA: a flexible large scale topic modeling package using variational inference in MapReduce, in: WWW, 2012, pp. 879–888.
[52] Y. Zhuang, W.-S. Chin, Y.-C. Juan, C.-J. Lin, A fast parallel SGD for matrix factorization in shared memory systems, ACM Recommender Systems, 2013.