MPI/OpenMP Hybrid Parallel Inference for Latent Dirichlet Allocation

Shotaro Tora
Graduate School of System Informatics
Kobe University
1-1 Rokkodai, Nada, Kobe 657-8501, Japan
tora@cs25.scitec.kobe-u.ac.jp

Koji Eguchi
Graduate School of System Informatics
Kobe University
1-1 Rokkodai, Nada, Kobe 657-8501, Japan
eguchi@port.kobe-u.ac.jp

ABSTRACT

In recent years, probabilistic topic models have been applied to various kinds of data including text data, and its effectiveness has been demonstrated. Latent Dirichlet Allocation (LDA) is one of the well-known topic models. Variational Bayesian inference or collapsed Gibbs sampling is often employed to estimate parameters in LDA; however, these inference methods require high computational cost for large-scale data. Therefore, high efficiency technology is needed for this purpose. In this paper, we make use of parallel computation technology for the sake of efficient collapsed Gibbs sampling inference for LDA. We assume to use a shared memory cluster (SMP cluster), which is widely used in recent years. In prior work of parallel inference for LDA, either MPI or OpenMP has been used alone. On the other hand, for a SMP cluster it is more suitable to adopt hybrid parallelization that uses message passing for communication between SMP nodes and loop directives for parallelization within each SMP node. In this paper, we developed a MPI/OpenMP hybrid parallel inference method for LDA, and achieved remarkable speedup under various settings of a SMP cluster.

Categories and Subject Descriptors

H.4 [Information Systems Applications]: Miscellaneous

General Terms

Experimentation, Performance, Design

Keywords

Topic modeling, inference, parallel computation

1. INTRODUCTION

For analyzing a large document collection, topic modeling approach is one of the most successful learning algorithms. The topic models are based on the idea that each document is generated from the mixture of the word distributions, each of which is called "topic". Latent Dirichlet Allocation (LDA)[4] is a well-known topic model. We can use the LDA model or its variants for not only text data but also image data[3, 6], network data[15, 1], and others.

However, inference for the LDA model on large-scale data brings significant challenges in terms of computation time and memory requirements. For this purpose, some speed-up techniques have been proposed via parallelization, such as approximate distributed inference for LDA (AD-LDA)[9] and asynchronous distributed learning algorithm (Async-LDA)[2].

The parallel computing architectures can be divided into three classes, distributed memory, symmetric multiprocessing (SMP), and SMP cluster. To optimize the performance and resources of these architectures, a promising approach is hybrid parallel computing, which use message passing for the communication between SMP nodes in a cluster and parallel computing based on shared memories in each node. We use Message Passing Interface (MPI) for the message passing between SMP nodes and Open Multi-Processing (OpenMP) for parallelization within each SMP node, for the inference of the LDA model.

In parallel computing using only MPI, each processor core communicate with each other by message passing, even though they belong to the same node, just like on distributed memory system. In the MPI/OpenMP hybrid parallel model, each processor core in a SMP node refers to data on a shared memory, and the only one processor core on each node communicate with each other. In general, it is efficient for processor cores to communicate via a shared memory than by message passing. When data are transferred by MPI function, such as 'MPI_Allreduce', the less number of processors take part in the communication, the shorter time it takes. Therefore, it is expected to improve the speed of learning the LDA model using MPI/OpenMP hybrid parallelization on SMP cluster

In this work, we propose MPI/OpenMP hybrid parallel inference for the LDA model, and demonstrate that it achieves significant speedup and keeps model estimation accuracy, when compared with both the case of using only MPI and the case of a single processor.

2. RELATED WORK

2.1 LDA

Latent Dirichlet Allocation (LDA) is a well-accepted topic model. It is based on the idea that each document is gener-

LDMTA '11 San Diego, California USA

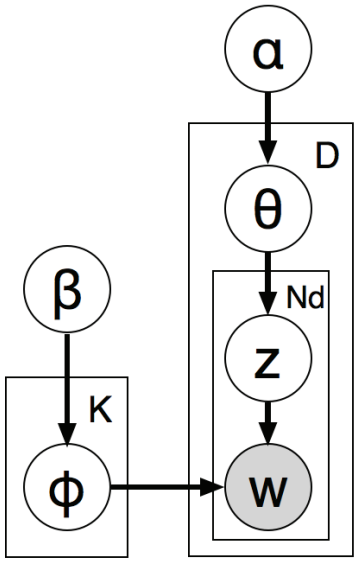

Figure 1: A graphical model representation of LDA

ated from a mixture of multinomial distributions over words, each of which is called a 'topic' and represents a cluster of words that co-occur across different documents. In this model, Dirichlet priors are assumed corresponding to both each document's topic multinomial and each topic's word multinomial. Figure 1 shows a graphical model representation of LDA, and the following are the process of generating documents. Here, α and β denote hyperparameters of the Dirichlet priors, and K, D, and N_d denote the number of topics, the number of documents, and the number of total words in document d.

1. For document d, multinomial parameters θ_d are drawn from Dirichlet prior distribution $Dir(\alpha)$.
2. For topic k, multinomial parameters ϕ_k are drawn from Dirichlet prior distribution $Dir(\beta)$.
3. For i-th word $w_{d,i}$ in document d:
   - topic $z_{d,i}$ is drawn from multinomial distribution $Mult(\theta_d)$
   - word $w_{d,i}$ is drawn from multinomial distribution $Mult(\phi_{z_{d,i}})$

The posterior distributions can be estimated using collapsed Gibbs sampling algorithm, which updates each topic assignment $z_{d,i}$ by sampling with the following full conditional posterior probability, in each iteration[5]:

$$p(z_{d,i} = k | w_{d,i} = v, W_{-(d,i)}, Z_{-(d,i)}) \propto (C^{doc}_{d,k} + \alpha) \frac{C^{word}_{k,v} + \beta}{\sum_{v'} C^{word}_{k,v'} + V\beta} \quad (1)$$

where C^{word} and C^{doc} stand for the word-topic count matrix and the document-topic count matrix, respectively. $W_{-(d,i)}$ denotes the document-word matrix in the document collection with the exclusion of $w_{d,i}$, and $Z_{-(d,i)}$ represents the corresponding topic assignments of $W_{-(d,i)}$.

The sampling procedure starts with random topic assignments, and then the posterior distributions given by the procedure above with a sufficient number of iterations will be converged according to the dependencies between random variables.

2.2 Fast inference methods for LDA

The computational complexity of collapsed Gibbs sampling is given by the number of topics multiplied by the vocabulary size in the document collection. There are some prior works attempting to improve the speed of inference for LDA model at various strategic points.

- Newman, et al.[9] proposed approximate distributed inference for LDA: AD-LDA and HD-LDA, using collapsed Gibbs sampling for distributed memory systems. AD-LDA achieves speedup without losing accuracy, although it is an approximate algorithm. HD-LDA is theoretically equivalent to estimating a mixture of LDA models; however, it suffers from high computational cost. Our inference method described in Section 3 can be positioned as an extension of AD-LDA.
- Yi Wang, et al.[12] implemented Parallel LDA (PLDA) on multiple machines by MPI or MapReduce. The algorithm of PLDA is equivalent to AD-LDA. They reported that MPI-PLDA was faster than MapReduce-PLDA, because the latter involves machine scheduling and disk I/O between iterations. PLDA can considerably reduce the total running time; however, it assumes that all processors are independent within and across nodes and requires global synchronization between them at each iteration. Therefore, the communication cost increases as the number of processors increases, in this case. We address this problem in this paper, assuming the use of a SMP cluster comprising multi-core processors.
- Asuncion, et al.[2] presented asynchronous distributed inference for LDA (Async-LDA). The data are distributed to processors, in each of which independently performs collapsed Gibbs sampling, but the processors communicate with each other in an asynchronous way. The global synchronization phase of AD-LDA in each Gibbs sweep (each iteration in Gibbs sampling) may force fast processors to wait for the slowest processor. In contrast, Async-LDA requires no global synchronization, and therefore each processor can start its next Gibbs sweep without having to wait for slower processors. Async-LDA can perform effectively in a distributed environment with heterogeneous machines having different specifications. However, this is not the case for the use of the machines having same specifications. In this paper, we focus on the conditions with a homogeneous environment.
- Masada, et al.[8] and Yan, et al.[13] made use of GPGPU for the parallel inference of LDA. Their focus was on the limit of memory size of GPGPU, which is different from our motivation of this paper.

3. MPI/OPENMP HYBRID PARALLEL INFERENCE FOR LDA

Newman et al.'s approximate distributed inference for LDA (AD-LDA)[9] and its implementations[12] can save considerable memory and time. However, it requires global synchronization at each iteration, and therefore the entire processing time is dominated by the communication overhead with the increase in the number of processors, thus forcing a limit in speedup. In this work, we attempt to improve the speed of LDA inference, keeping inference accuracy. In this section, we first briefly review general computing models in parallel and distributed environments, and then describe hybrid parallel inference for LDA.

3.1 Parallel and distributed computing models

We below summarize MPI, OpenMP, and MPI/OpenMP hybrid parallelization, which supplies the low-level services necessary for parallel computing.

3.1.1 MPI

Message Passing Interface (MPI) is a popular protocol for programming parallel computing. In this work, MPI is used for the inter-node message-passing communications in a SMP cluster.

3.1.2 OpenMP

Open Multi-Processing (OpenMP) supports shared-memory parallel programming. We use it for shared-memory parallelization within each node of a SMP cluster.

3.1.3 MPI/OpenMP hybrid parallelization

In terms of data communication, shared-memory parallelization is faster than message passing. The less number of processors take part in the message passing communication, the shorter time it takes. Therefore, MPI/OpenMP hybrid programming model sometimes achieves substantial speedup on SMP clusters[10]. We discuss in the following section how to speed up the inference of LDA using the MPI/OpenMP model, keeping inference accuracy.

3.2 Inter-node parallelization

3.2.1 Distributing and synchronization of topic-word distributions

The inter-node parallel procedure can be achieved in the manner similar to AD-LDA[9], as follows. D documents are distributed over N nodes, assigning $D_n = D/N$ documents to each node. AD-LDA algorithm partitions document counts $W = \{\mathbf{w}_d\}_{d=1}^{D}$ into $\{W_{|1}, ..., W_{|N}\}$ and corresponding topic assignments $Z = \{\mathbf{z}_d\}_{d=1}^{D}$ into $\{Z_{|1}, ..., Z_{|N}\}$, where $W_{|n}$and $Z_{|n}$ exist only on node p. Document-specific counts C^{doc} are likewise distributed; however, every node maintains its own copies of word-topic counts C^{word}. We denotes node-specific counts as $C_{|n}^{doc}$. Moreover, $C_{|n}^{word}$ is used to temporarily store word-topic counts accumulated from topic assignments to local documents on each node.

In each Gibbs sweep (each iteration in Gibbs sampling), $Z_{|n}$ is updated on each node n by sampling every $z_{d,i|n} \in Z_{|n}$ from the approximate posterior distribution, and each node updates $C_{|n}^{doc}$ and C^{word} according to the new topic assignments. After each Gibbs sweep, each node samples and updates word-topic counts of its local documents $C_{|n}^{word}$ and uses Allreduce operation to reduce and broadcast the new C^{word} to all nodes.

$C_{|n}^{word}$ denotes the result of N separate word-topic count matrices estimated from the separate data. In particular, $\sum_{word} C_{|n}^{word} = W$, where W is the total number of words across all nodes. After node n samples and updates topic assignments $z_{|n}$, we modify both $C_{|n}^{doc}$ and $C_{|n}^{word}$. In order to merge the per-node counts, we perform the global update using a Reduce-Scatter operation:

$$C^{word} \leftarrow C^{word} + \sum_{p} (C_{|n}^{word} - C^{word}),$$
$$C_{|n}^{word} \leftarrow C^{word} \quad (2)$$

where $C_{|n}^{word}$ denotes the counts that all nodes started with before the Gibbs sweep.

3.2.2 Synchronization of document-topic distributions

Following [9], document data are distributed over multiple nodes for speeding up LDA inference. In the case of [9], the document data are divided in units of documents. For comparison, we divide document data in two ways: in units of documents and in units of words.

When document data are divided in units of documents, we have to perform inter-node communication and synchronization only for $C_{|n}^{word}$ in each Gibbs sweep, using Eq.(2). $C_{|n}^{doc}$ is not necessary for the transfer across nodes or processors. Therefore, we can reduce the cost for communications, in this case.

When document data are divided in units of words, the number of words on each node is roughly equal, and therefore we can reduce the cost for synchronization of count matrices. In this case, we have to perform inter-node communication and synchronization for both $C_{|n}^{word}$ and $C_{|n}^{doc}$ using Eqs.(2) and (3), respectively.

$$C^{doc} \leftarrow C^{doc} + \sum_{p} (C_{|n}^{doc} - C^{doc}),$$
$$C_{|n}^{doc} \leftarrow C^{doc} \quad (3)$$

3.3 Intra-node parallelization

D_n documents allocated to node n are distributed over P processors inside the node, and then $D_{np} = D_n/P$ documents are taken care of by each processor. We partition $W_{|n}$ into $\{W_{|n1}, ..., W_{|nP}\}$ on node n; however, $Z_{|n}$ is not divided, but shared by all the processors inside the node. Moreover, each processor handles the original $C_{|n}^{word}$, not the copy of it. Each processor p samples topic assignment $z_{d,i|n} \in Z_{|n}$ to update $Z_{|n}$, and then updates $C_{|n}^{doc}$ and $C_{|n}^{word}$ according to the new topic assignment.

As above, the intra-node parallel algorithm using multiple processors is not approximate inference in the sense that we do not use copies of word-topic counts. However, the inference result is not the same as result of serial inference using a single processor, since the intra-node parallel algorithm divides the document data into $D_{np} = D_n/P$ and thus the sampling order is different from that of the serial algorithm.

Figure 2 illustrates the communication pattern of AD-LDA. When AD-LDA algorithm is performed independently using all processors within and across nodes, the number of processors that take part in the inter-node communications is NP, where N and P denote the number of nodes and the number of processors of each node, respectively. To speed up the inference, we parallelize the inference procedure inside each node using OpenMP. As shown in Figure 3, the

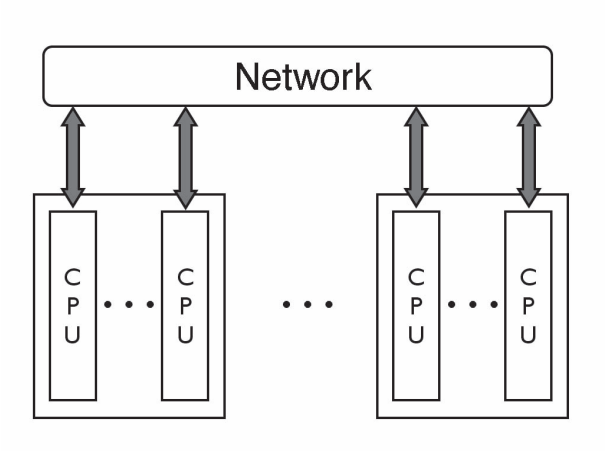

Figure 2: Communications between all processors in the case of AD-LDA

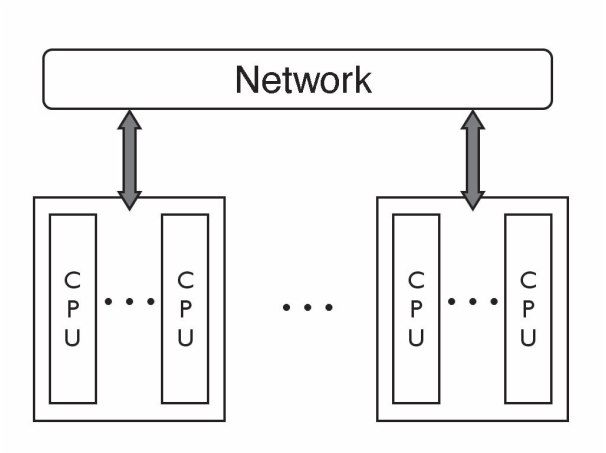

Figure 3: Communications between nodes in the case of the hybrid inference

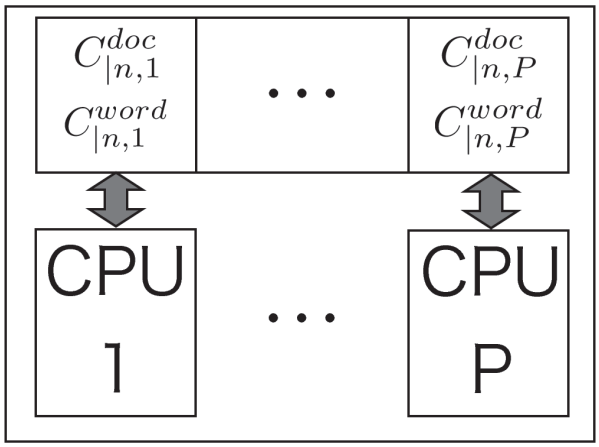

Figure 4: Data allocation inside each node in the case of AD-LDA

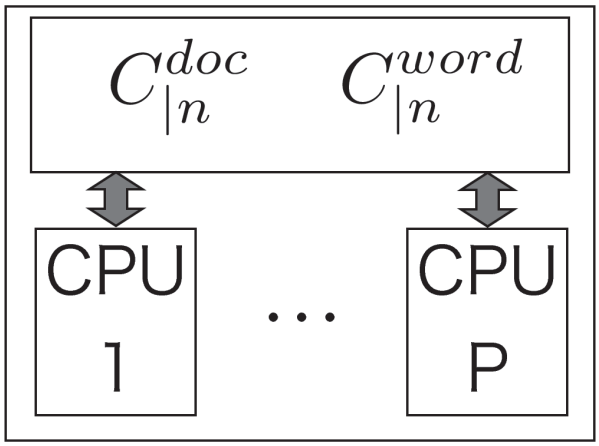

Figure 5: Data allocation inside each node in the case of the hybrid inference

number of processors that take part in the inter-node communications is N in the MPI/OpenMP hybrid inference for LDA, even if all the processor cores of all nodes are used.

This hybrid inference algorithm uses copies of word-topic counts on each node, while AD-LDA independently using all processors across nodes requires the copies of word-topic counts for each processor. As shown in Figures 4 and 5, the hybrid inference algorithm can save considerable memory when applied to large document data.

4. EXPERIMENTS

To evaluate MPI/OpenMP hybrid parallel inference for LDA, we measure inference speed and test-set perplexity. For each test document, we divided its words into training and test sets. We estimated each document's topic distribution $\theta_{d,k}$ and each topic's word distribution $\phi_{k,w}$ using the training set, and then computed test-set perplexity using $\theta_{d,k}$ and $\phi_{k,w}$.

4.1 Settings

In our experiments, we use the hybrid inference algorithm with increasing degree of parallelization, compared with other conventional inference algorithms: LDA[5] and AD-LDA[9]. We used a 15-node SMP cluster comprising 8 processor cores for each node, changing the number of processor cores for the use of inference as $P \in \{1, 2, 4, 6, 8\}$ for each node. When $P = 1$, AD-LDA parallelizes over 15 nodes comprising one processor core for each node, and therefore AD-LDA is equivalent to the hybrid inference algorithm in that case. For the MPI implementation, we used MPICH2 in both cases of AD-LDA and the hybrid inference algorithm. We set the number of topics $K = 400$, and Dirichlet hyperparameters $\alpha = 50/K$ and $\beta = 0.1$ [11].

4.2 Data set and evaluation metrics

In our experiments, we used The New York Times news article data[1]. A summary of this data set is shown in Table 4, where D is the number of documents, W is the vocabulary

[1] http://archive.ics.uci.edu/ml/datasets/Bag+of+Words

Table 1: Summary of the NYTimes data

	D	V	W (approx.)
NTTimes	300000	102660	100,000,000

size and N is the approximate total number of words.

We measured inference speed and test-set perplexity, for evaluation. The test-set perplexity is given by:

$$P(\mathbf{w}^{test}) = \prod_{d,i} \sum_{k} \theta_{d,k} \phi_{k,w_{d,i}^{test}} \tag{4}$$

$$\theta_{d,k} = \frac{C_{d,k}^{doc} + \alpha}{\sum_{k'} C_{d,k'}^{doc} + K\alpha} \tag{5}$$

$$\phi_{k,v} = \frac{C_{k,v}^{word} + \beta}{\sum_{v'} C_{k,v'}^{word} + V\beta} \tag{6}$$

4.3 Results

Figure 6 demonstrates the comparison of speed-up performance of AD-LDA and the MPI/OpenMP hybrid inference, in the case that document data are divided in units of documents. To evaluate the speed-up performance, we use the conventional serial inference of LDA as the baseline. As shown in these results, our hybrid inference achieved the successful speed-up performance as the degree of parallelization increases, even when parallelization degree is 120. AD-LDA also achieved the performance as the degree of parallelization increases, to some extent; however, the performance gets saturated around 60-way parallelization.

For reference, Figure 7 demonstrates the comparison of the two inference methods, in the case that document data are divided in units of words and thus the number of words on each node or processor core is roughly equal. As results, our hybrid inference shows a performance advantage than AD-LDA in this case, as well. Not surprisingly, the overall performance in this case was not so good as in the case that document data are divided in units of documents, as shown previously in Figure 6. This is because when document data are divided in units of documents, it can reduce communication cost, avoiding the transfer of document-topic counts across nodes or processor cores. On the other hand, when document data are divided in units of words, it can reduce synchronization cost; however, the loss of communication cost is generally more critical than that of synchronization cost.

We will present further experimental results in the following paragraphs. Tables 2 and 3 show the detailed results when document data are divided in units of documents. In these tables, you can see that the hybrid inference can reduce communication cost, compared with AD-LDA. Moreover, the hybrid inference keeps constant communication time, even when the number of processor cores increases. As we mentioned previously, our hybrid inference achieved the successful speed-up performance as the degree of parallelization increases; however, AD-LDA's performance gets saturated under a highly parallelized setting. This performance gap is caused by the fact that the inter-node communications with the shared-memory multi-processors can reduce the cost than the inter-processor communications via message passing.

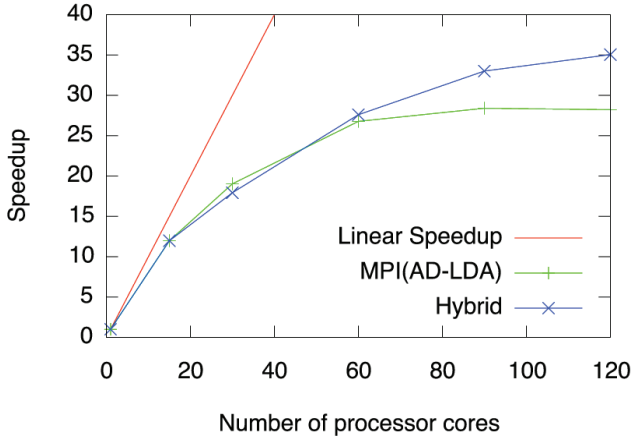

Figure 6: Speed-up performance for the number of processor cores (in the case that document data are divided in units of documents)

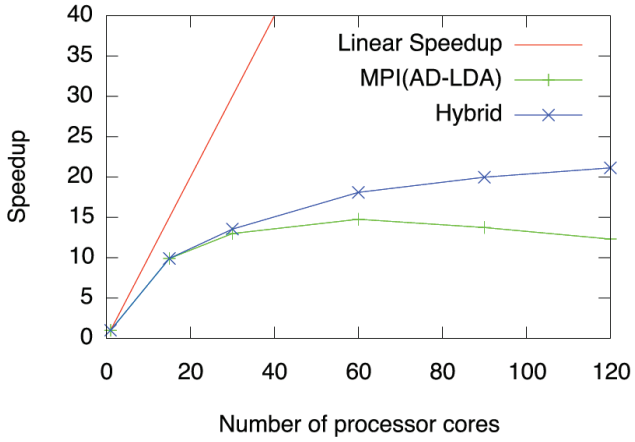

Figure 7: Speed-up performance for the number of processor cores (in the case that document data are divided in units of words)

For reference, Tables 4 and 5 demonstrate the detailed results when document data are divided in units of words and thus the number of words on each node is roughly equal. The hybrid inference keeps constant communication time, as shown in Table 5. However, the communication time is much longer, compared with the case that document data are divided in units of documents, as in Table 3. The synchronization time is greatly reduced; however, it is much less than the communication time.

In Figures 8 and 9, we demonstrate the test-set perplexity for LDA, AD-LDA and the hybrid inference. From these results, you can see that the difference in perplexity between these models is very small. Figure 10 shows that the test-set perplexity at the 500th iteration, changing the number of processor cores used. As can be seen in this figure, the test-set perplexity was changed only slightly even when the number of processor cores is increased, and the test-set per-

Table 2: Speed-up performance of AD-LDA (in the case that document data are divided in units of documents)

number of cores	computation time(sec)	comm. time(sec)	sync. time(sec)	total time(sec)	speedup ratio
1	192708.00	0.00	0.00	192708	1.00
15	13244.86	1828.53	1018.60	16192	11.90
30	6861.50	2417.26	848.23	10127	19.02
60	3816.08	2620.26	760.65	7197	26.77
90	2913.15	3112.77	758.06	6784	28.40
120	2927.59	3224.45	674.95	6827	28.22

Table 3: Speed-up performance of the hybrid inference (in the case that document data are divided in units of documents)

number of cores	computation time(sec)	comm. time(sec)	sync. time(sec)	total time(sec)	speedup ratio
1	192708.00	0.00	0.00	192708	1.00
15	13244.86	1828.53	1018.60	16192	11.90
30	7968.13	1836.53	943.33	10748	17.92
60	4466.06	1785.33	730.60	6982	27.60
90	3361.80	1783.46	691.73	5837	33.01
120	3022.80	1797.06	678.13	5498	35.05

Table 4: Speed-up performance of AD-LDA (in the case that document data are divided in units of words)

number of cores	computation time(sec)	comm. time(sec)	sync. time(sec)	total time(sec)	speedup ratio
1	192708.00	0.00	0.00	192708	1.00
15	14103.41	5067.93	284.66	19456	9.90
30	7840.66	6735.10	257.23	14833	12.60
60	4448.58	8388.41	215.00	13052	14.76
90	4698.08	9121.44	204.47	14024	13.74
120	5253.39	10205.44	194.15	15653	12.14

Table 5: Speed-up performance of the hybrid inference (in the case that document data are divided in units of words)

number of cores	computation time(sec)	comm. time(sec)	sync. time(sec)	total time(sec)	speedup ratio
1	192708.00	0.00	0.00	192708	1.00
15	14103.41	5067.93	284.66	19456	9.90
30	8902.00	5071.26	254.73	14228	13.12
60	5322.20	5086.33	239.46	10648	17.25
90	4271.13	5163.26	210.60	9645	18.97
120	3861.13	5062.13	193.73	9117	21.13

plexity with the hybrid inference was even better than that of the standard LDA in any cases.

5. CONCLUSIONS

In this paper, we demonstrated that MPI/OpenMP hybrid parallel algorithm for the inference of Latent Dirichlet Allocation can substantially improve the inference speed by reducing the communication cost, keeping inference accuracy. We also investigated the ways of separating document data for parallelization.

Experiments with larger data sets are currently under investigation. Further employing algorithmic improvements[14] and pipeline processing or other scheduling techniques[7] is also left for the future work.

Acknowledgments

We thank Dr. Tomio Kamada and the anonymous reviewers for valuable discussions and comments. This work was supported in part by the Grants-in-Aid for Scientific Research (B) (#20300038 and #23300039) from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

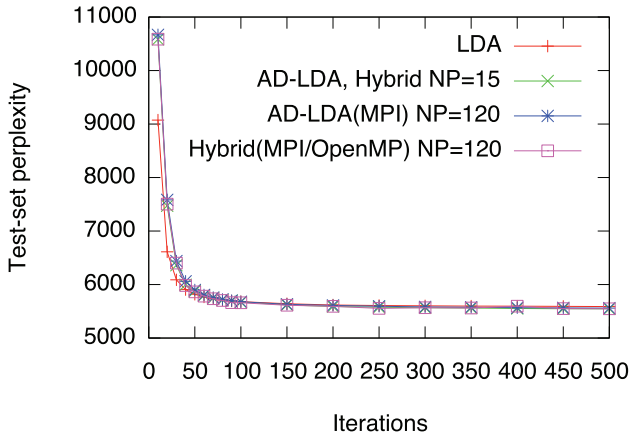

Figure 8: Test-set perplexity for each 50 iterations (in the case that document data are divided in units of documents)

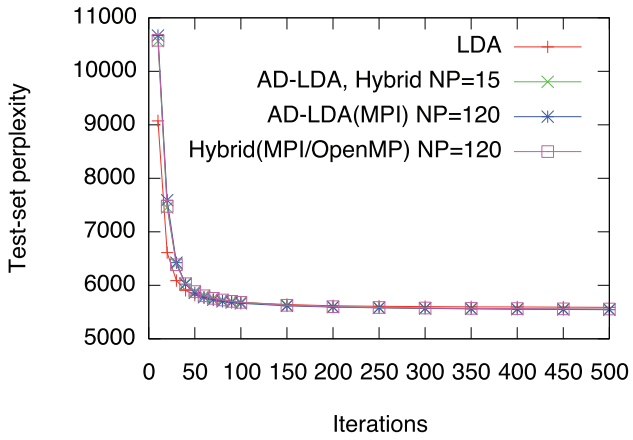

Figure 9: Test-set perplexity for each 50 iterations (in the case that document data are divided in units of words)

6. REFERENCES

[1] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing. Mixed membership stochastic blockmodels. Journal of Machine Learning Research, 9:1981–2014, 2008.

[2] A. Asuncion, P. Smyth, and M. Welling. Asynchronous distributed learning of topic models. In Advances in Neural Information Processing Systems, volume 21, pages 81–88. MIT Press, 2008.

[3] D. M. Blei and M. I. Jordan. Modeling annotated data. In Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, pages 127–134, Toronto, Canada, 2003.

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

[5] T. L. Griffiths and M. Steyvers. Finding scientific topics. Proceedings of the National Academy of Sciences of the United States of America, 101:5228–5235, 2004.

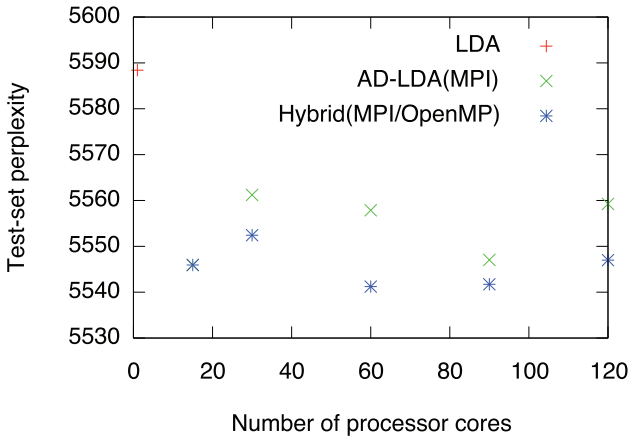

Figure 10: Test-set log-likelihood for the number of processor cores (in the case that document data are divided in units of documents)

[6] F.-F. Li and P. Perona. A Bayesian hierarchical model for learning natural scene categories. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 2, pages 524–531, San Diego, California, USA, 2005.

[7] Z. Liu, Y. Zhang, E. Y. Chang, and M. Sun. PLDA+: Parallel latent Dirichlet allocation with data placement and pipeline processing. ACM Transactions on Intelligent Systems and Technology, 2(3):26:1–26:18, 2011.

[8] T. Masada, T. Hamada, Y. Shibata, and K. Oguri. Accelerating collapsed variational Bayesian inference for latent Dirichlet allocation with Nvidia CUDA compatible devices. In Next-Generation Applied Intelligence: 22nd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2009, Tainan, Taiwan, volume 5579 of Lecture Notes in Computer Science, pages 491–500. Springer, 2009.

[9] D. Newman, A. Asuncion, P. Smyth, and M. Welling. Distributed inference for latent Dirichlet allocation. In Advances in Neural Information Processing Systems, volume 20, pages 1081–1088. MIT Press, 2007.

[10] R. Rabenseifner, G. Hager, and G. Jost. Hybrid MPI/OpenMP parallel programming on clusters of multi-core SMP nodes. In Proceedings of the 2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing, pages 427–436, Weimar, Germany, 2009.

[11] M. Steyvers and T. Griffiths. Handbook of Latent Semantic Analysis, chapter 21: Probabilistic Topic Models. Lawrence Erbaum Associates, Mahwah, New Jersey and London, 2007.

[12] Y. Wang, H. Bai, M. Stanton, W.-Y. Chen, and E. Y. Chang. PLDA: Parallel latent Dirichlet allocation for large-scale applications. In Proceedings of the 5th International Conference on Algorithmic Aspects in Information and Management, pages 301–314, San Francisco, California, USA, 2009.

[13] F. Yan, N. Xu, and Y. Qi. Parallel inference for latent Dirichlet allocation on graphics processing units. In Advances in Neural Information Processing Systems, volume 22, pages 2134–2142. MIT Press, 2009.

[14] L. Yao, D. Mimno, and A. McCallum. Efficient methods for topic model inference on streaming document collections. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 937–946, Paris, France, 2009.

[15] H. Zhang, C. L. Giles, H. C. Foley, and J. Yen. Probabilistic community discovery using hierarchical latent Gaussian mixture model. In Proccedings of the 22th Association for the Advancement of Artificial Intelligence Conference (AAAI 2007), pages 663–668, Vancouver, Canada, 2007.