A Scalable Approach for Topic Modeling with R

Tiffany A. Connors
Texas State University
San Marcos, TX, USA
tiffanyaconnors@gmail.com

Ritu Arora
Texas Advanced Computing Center
Austin, TX, USA
rauta@tacc.utexas.edu

ABSTRACT

Topic Modeling (TM) helps in automatically classifying documents under different topics, and is especially useful for exploring a large corpus of documents to discover new relationships. The R programming language has a TM library that is easy to install and use. However, due to its interpreted nature, the performance of TM code written in R is poor as compared to the same code rewritten in C/C++/Fortran. Despite its poor performance, R is a high-productivity language that is commonly used by non-traditional High Performance Computing (HPC) users to do TM and other similar data analyses. Many such users do not have access to expertise for rewriting their R code in C/C++/Fortran but have large datasets to analyze. With TM as an example, we demonstrate that such end-users can reduce the time-to-results (sometimes, up to a factor of 23) by running their R code in High-Throughput Computing (HTC) mode on HPC resources.ă

1. INTRODUCTION

TM uses probabilistic models to identify patterns of word use within a corpus. It helps in analyzing large volumes of text and discovering underlying relationships. Topics can be used to annotate, organize, summarize, and search document collections. Latent Dirichlet Allocation (LDA)[1] is a hierarchical Bayesian model that discovers semantic topics within text corpora. In LDA, each document within a collection is represented as a mixture of topics and hence, can be assigned under multiple classes of topics. TM can be performed using R packages. Additional packages in R can then be used for visual exploration of the topics. R is available on several High Performance Computing (HPC) platforms at open-science data centers (e.g., the Stampede supercomputer at the Texas Advanced Computing Center). A limitation of R is that the performance of R code can be poor as compared to C/C++/Fortran implementations of the same algorithm. This is due to R being an interpreted language. While there are some parallel packages available for R, many popular packages (like the package for TM) do not have parallel implementations.

As it is a high-productivity language, domain-scientists and several non-traditional HPC users commonly use R. Many such users may not have access to expertise for rewriting R code in C/C++/Fortran. Users with R code written before the emergence of parallel packages may not have time or expertise to update their code to take advantage of the parallel implementations when they become available in future. However, such users may have extensive needs for analyzing large datasets (e.g., geoscientists), and can benefit from large amounts of memory available on compute nodes of modern HPC platforms. HTC can enable such users to transition to HPC while leveraging the high-productivity of R and the investments made in their existing codebase.

1.1 High Throughput Computing (HTC)

HTC involves running multiple copies of a serial application concurrently on multiple cores and nodes of a platform such that each copy of the application uses different input data or parameters. By running R scripts in HTC mode, users can take advantage of the state-of-the-art HPC platforms and large-scale storage resources without having to rework their code. HTC mode can be used on the Stampede supercomputer at TACC by utilizing the Launcher tool that wasădeveloped in-house at TACC[2]. The Launcher requires the user to supply a SLURM jobăscript (named "launcher.slurm")ăand aăfile containing the list of commands to run concurrently (named "paramlist"). We have developed scripts that can automatically generate these files and do load-balancing across multiple nodes of the HPC platform.

The main script, "HTC_TopicModeling.sh", determines the number of subdirectories within the user specified directory. The scripts then adds the command for analyzing each subdirectory to the "paramlist" file. The number of computational cores and nodes to use for the HTC job is determined by the number of tasks in the "paramlist" file.ăIf the number of tasks is greater than 16 (max number of cores/node), additional nodes are requested. A custom "launcher.slurm" file is then generated based on the needs of the job.

2. EXPERIMENTAL DESIGN

To demonstrate the effectiveness of our approach in accelerating the time-to-results while using the R programs, we wrote a customizable R script for TM. We selected three publicly available datasets for TM - BBC, BBCSport[3], and NSF Research Award Abstracts[4] - using the Stampede supercomputer at TACC. The BBC dataset consists of 2,225 text documents from the BBC news website. The BBCSport dataset contains 737 documents related to sports news. The NSF Research Award Abstracts dataset consists of 129,000 abstracts describing NSF awards. In our study, we used a subset of 10,097 abstracts from 1990.

Our R script for TM utilizes the LDA function in the topicmodels[5] package to perform TM on a collection of documents. The script performs pre-processing, such as stemming and removing stop-words. Next, the document term

matrix is created and LDA is performed. CSV files of the word frequencies and results of the LDA TM are created. Lastly, our R script generates visualizations using the wordcloud[6] and LDAvis[7] packages.

3. RESULTS

By utilizing HTC, we are able to improve the runtime performance by nearly a factor of 3 for both BBC and BBCSports datasets. For the NSFAwards dataset, utilizing HTC improved the runtime by more than a factor of 23. The results are shown in Figure 1 and Figure 2. The runtimes for each serial and HTC run of the datasets are shown in Table 1.

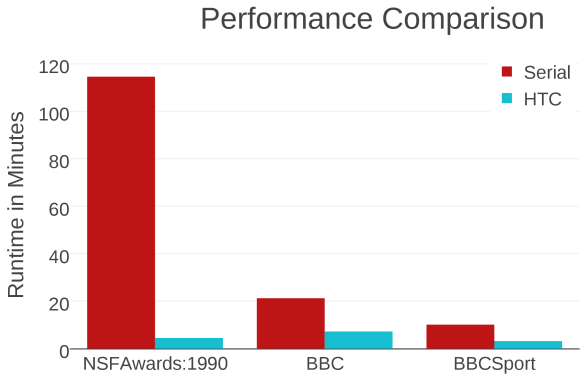

Figure 1: Comparison of serial and HTC runtimes by dataset.

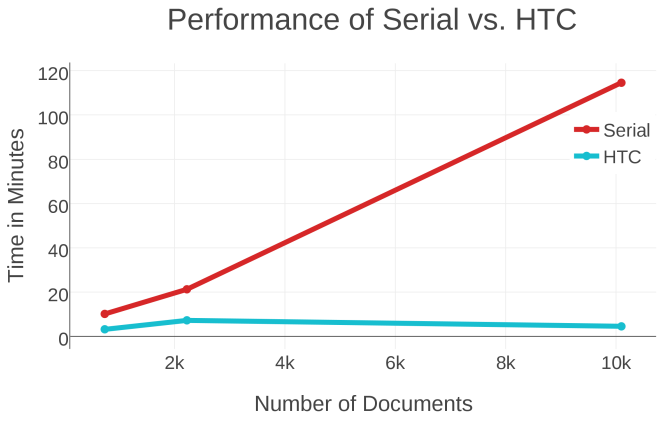

Figure 2: Performance of Serial vs. HTC in relation to the number of documents.

Table 1: Datasets used for TM

Dataset	Docs	Serial Time	HTC Time	Cores
NSFAwards:1990	10,097	114m 57s	4m 50s	42
BBC	2,225	21m 25s	7m 23s	5
BBCSport	737	10m 12s	3m 18s	5

4. CONCLUSIONS

Through this project, we have demonstrated an approach to lower the adoption barriers to HPC resources by leveraging HTC. While we focused on TM in this project, the underlying approach of running existing R scripts using HTC is scalable and can be generalized for course-grained data parallelism of other R scripts for data analyses as well.

5. REFERENCES

[1] D. Blei, A. Ng, and M. Jordan. 2003. Latent Dirichlet Allocation. J. Mach. Learn. Res. 3 (March 2003), 993-1022.

[2] Stampede User-Guide, website accessed on October 10, 2016 https://www.tacc.utexas.edu/researchdevelopment/tacc-software/the-launcher

[3] D. Greene and P. Cunningham. Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering, Proc. ICML 2006.

[4] Lichman, M. 2013. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[5] CRAN - Package topicmodels, website accessed on October 10, 2016 https://cran.r-project.org/web/packages/topicmodels/index.html

[6] CRAN - Package wordcloud, website accessed on October 10, 2016 https://cran.r-project.org/web/packages/wordcloud/index.html

[7] C. Sievert and K. Shirley. Ldavis: A method for visualizing and interpreting topics. In 2014 ACL Workshop on Interactive Language Learning, Visualization, and Interfaces, Baltimore, June 2014.