Scalable Text Clustering on Global Grids

YuZheng Zhai[1], XingChen Chu[2], Arthur Hsu[1], Saman K Halgamuge[1] and Rajkumar Buyya[2]

Dynamic System and Control Group[1]
Department of Mechanical and Manufacturing Engineering

Grid Computing and Distributed Systems (GRIDS) Laboratory[2]
Department of Computer Science and Software Engineering

The University of Melbourne, Australia

y.zhai@pgrad.unimelb.edu.au, {xcchu, alhsu, saman, rbuyya}@unimelb.edu.au

Abstract. Processing the ever increasing amount of textual information is beyond the capability of typical local computing resources. The emergence of grid computing technology enables the sharing and aggregation of computational resources, thus facilitates the rapid analysis of information. In this paper we present the use of computational grids in the domain of clustering large scale text collections. A novel two-level GSOM is used to parallelise the clustering task. We observe a performance improvement of 10 times speedup over the traditional method executed on a single computational node.

1 Introduction

The emergent technology of grid computing [3] enables the sharing of geographically distributed heterogeneous resource to solve computationally intensive problems in various disciplines. Many projects have already been developed using the grid in areas such as weather forecasting, finical modeling and earthquake simulation, to access computational resources, data collections and remote instrumentations. Grid computing augments traditional research methodologies to advance to the next generation e-Research, which encourages collaboration between complimentary domains and enables researchers to perform more creatively and efficiently. It also shows great promises as the platform for large-scale applications.

The ever increasing amount of textual information available on the internet has made textual data clustering algorithms more appealing and promising. However, the demand for enormous processing power and resources to deal with these billions of documents is beyond the capability of typical local computing resources. Therefore, scalability has become an important issue. Some works have been done in the natural language processing domain that shows the benefit of using Grids in data-intensive applications [12,13]. It is clear that text clustering applications can also benefit from the grid to improve scalability, if we can parallelise the clustering task and execute them on the Grid.

The motivation of this work arises from the Australian Research Council funded E-Research project on “Collection, Sharing, Visualisation and Analysis of locally

gathered information from geographically remote areas vulnerable to tidal waves", where a proposed website will be open for public as a portal to collect substantiated reports of strange behavior in animals. These reports will then been analyzed, in conjunction with other source of information such as news articles, seismic data, etc to flag alert for further attention. The number of reports and news articles will grow larger as time passes. Therefore a scalable text mining technique combined with the capability of taking advantage of the available grid resources will be very useful in discovering valuable information from the rapidly expanding database.

In the previous studies [11], we proposed a document clustering approach based on the WEBSOM method using the scalable two-level Growing Self-Organising Maps (GSOM), which provides us the prospect of utilizing Grid resources to cluster massive text data sets efficiently. Our simulation show potentially significant improvement on the processing speed compare to the same process on a single computational node. In this paper, we carry out the experiment using the Gridbus Resource Broker [14] to execute the programs on heterogeneous grid resources to further evaluate its performance.

The rest of the paper is organized as follows. Section 2 presents an overview of the clustering algorithm and the grid middleware used. Section 3 describes the architecture, data set and evaluation method employed in the experiment. Results and comparisons are presented in Section 4. Conclusion and future research efforts are given in Section 5.

2 Background

2.1 Scalable text mining with GSOM

Dynamic Self-Organising Maps is an unsupervised neural network model that maps high dimensional input into a low dimensional topology such that similar clusters are close to each other on the map [4, 8, 10]. It can grow into different shapes and sizes corresponding to the input. GSOM starts with a small number of nodes and then goes through one growing phase, a rough tuning phase and a fine tuning phase. The rate of the growth and thus the final map size, is controlled by a Spread Factor (SF) that takes value between 0 and 1 (1 gives maximum growth) [9]. A detailed explanation and study of the choice of SF in textual clustering can be found in [11].

WEBSOM [5, 6] is an application of SOM that offers an alternative way to encode the text document rather than the traditional "bag of words" approach [7]. It uses the word category maps that utilize the contextual information to group similar words. These word context vectors are mapped onto a two-dimensional grid using the GSOM to form the word category map. Documents are then encoded as vectors of histogram of the word clusters formed earlier. These vectors serve as input to another GSOM, which produces the document map that enables the visualisation of clusters.

Our scalable method splits the initial single growing phase in GSOM into two separate growing phases. While the first growing phase is performed on a single computational node, the outcome is used to initiate the second growing phase running concurrently on different computational nodes, together with the remaining two

tuning phases. This is intended to first obtain an abstract and rapid grouping of the entire input data during the first growing phase by using a low spread factor with GSOM. It produces a feature map that has a few condensed nodes with high level separation between them. The data obtained within each node will then be sent to different computers and refined independently. They will be processed by another GSOM with a slightly higher spread factor in order to achieve finer clustering and follows the normal procedure of one growing and two tuning phases, as shown in Figure 1. All the resulting outputs will be passed back to combined together for further processing.

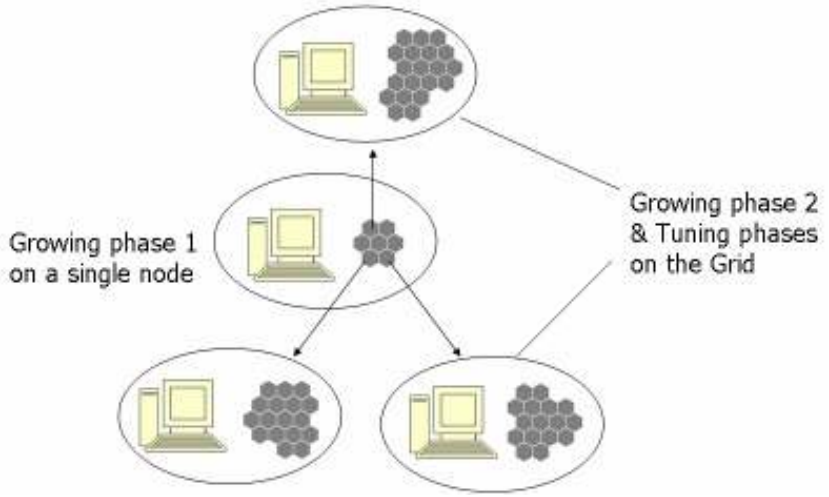

Figure 1. Two stages of scalable GSOM

3 Grid Architecture for scalable GSOM-based Analysis

The Grid environment addresses the problem of scalability for our GSOM based text mining with massive data sets, which currently largely affects the performance and productivity of the mining method. By targeting this research bottleneck, the mining results which traditionally take hours of waiting time will be obtained quicker. In order to build a Grid environment, middleware tools are needed to support various services such as security, uniform access, resource management, job scheduling, application composition and QoS. There are a lot of tools and software available to help establishing computational Grids including Globus, Condor, and also some resource management system such as Sun N1 Grid Engine (SGE) or Portable Batch System (PBS). However, we do not need to care about the underlying Grid systems been used when we dispatch our mining tasks to the resources, as the broker provides plug-ins to take care of different middleware.

The Gridbus Resource Broker is a user-level middleware that simplifies and optimizes the development and deployment of Grid applications. Moreover, it

provides a layer of abstraction by hiding the Grid resource specific details and providing application programming interfaces (API) for most generic functions needed by Grid applications. It can be used for composition and formulation of textual mining tasks and for deployment them on heterogeneous platforms in Global Grid.

The high-level system architecture used in this experiment is show in Figure 2 below. The documents are preprocessed and feed into the clustering program, which groups the entire document collections into a number of segments. Then a parameter sweep model is used which treats each segment as an input parameter to the clustering program to generate the application description file. This is then provided to the Gridbus Broker, which discovers available resources, composes the application, creates jobs, transfers the input data and executes them the remote resources. The results are retrieved after execution and combined together for post-processing to obtain the final results.

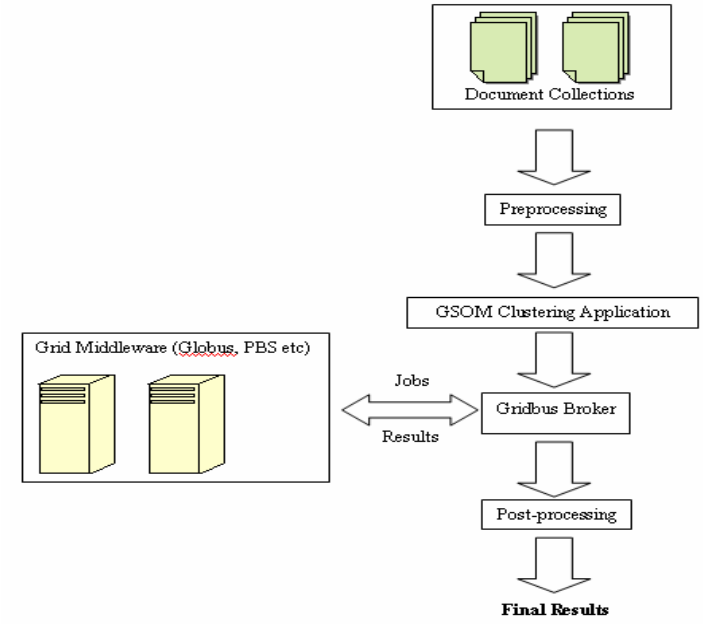

Figure 2. System architecture

4. Application Composition and Analysis on Global Grids

4.1 Document Collections

The document collection is composed of a subset of the forum articles obtained from the Usenet discussion group, which has been employed in WEBSOM studies in [1, 2],

as well as in our previous simulation [11]. This collection of documents is particularly interesting not only because it closely resembles the texts of the intended application, it is but also regarded as a challenging text collection since all the documents were written in a very informal language that often hold all sorts of colloquial words and produces an larger vocabulary than properly written news articles. The collection contains 2,000 articles in total and serves as the benchmark for measuring the execution time.

4.2 Preprocessing

The documents were preprocessed to remove any non-textual content such as HTML tags, header information and user signatures, which results in an average text length of 200 words for the second data set. After removing from a list of 385 stop words, the vocabulary contains 26,869 words (both base and inflected forms). The words that appear less than 5 times and more than 500 times, which are regarded as less useful in distinguishing between clusters, were also removed and the final vocabulary consists of 5707 words.

4.3 Application Composition

The GSOM clustering tasks were formulated as a parameter sweep application using XPML (XML-based Parametric Modeling Language) provided by the Gridbus Broker. The application processes each text cluster, which were generated in the first run of GSOM and is represented by the parameter "id". Figure 3 shows an example of XPML file that describes the analysis operation. A total of 35 clusters in compressed file format were processed in the experiment. The files were distributed into a directory tree and a shell script linked the files into one directory with standardized naming convention file[$id].tar.gz. The job tarball is transferred to the remote node and uncompressed there. The clustering program is then executed on the remote nodes and the results are transferred to the broker host on completion.

The four key parts of the entire XPML code shown in Figure 2 are discussed below:

- The parameter element defines the parameter-sweep, a variable "id" that varies from 0 to 34 steps. For each value, a new job will be created and assigned a task.
- The copy command with the data source labeled as "local" and destination labeled as "node"; "local" refers to a resource from which the broker launches the executions and "node" refers a remote resource selected by the broker's scheduling algorithm while job execution. This copy command is called stage-in which effectively transfers the required data including the input data and executable program to the Grid resources. The variable id is used with a $ prefix which notifies the broker to substitute the variable with a value from 0 to 34.
- Part 3 consists of several execute commands inside the task element, which describes the application execution workflow occurred on the Grid resources. Those execute commands will executed sequentially to carry out the final result. The execution occurred on each resource will firstly uncompress the input data

file, and then run a java program which is used to calculate the result. The variable substitution has also been applied for each data clusters we generated.

- The last copy command inside the task element is called the stage-out which is responsible for collecting results from various Grid resources and transferring them back to the user's workstation. The destination is not restricted to the local node where the Gridbus broker is running; it can also be a third party node which might to perform post-process against the result.

```
<xpml xmlns="http://schemas.gridbus.org/xpml/2006/01/xpml">
  <parameter domain="range" name="id" type="integer">
    <range from="0" interval="1" to="34" type="step"/>
  </parameter>
  <task>
    <copy>
      <source file="file$id.tar.gz" location="local"/>
      <destination file="file$id.tar.gz" location="node"/>
    </copy>
    <execute>
      <command value="gunzip file$id.tar.gz"/>
    </execute>
    <execute>
      <command value="tar -xvf file$id.tar"/>
    </execute>
    <execute>
      <command value="java -jar NNTool.jar wmap$id.tdd 0.3 $id.txt"/>
    </execute>
    <copy>
      <source file="$id.txt" location="node"/>
      <destination file="output.$jobname" location="local"/>
    </copy>
  </task>
</xpml>
```

Figure 3. XPML application description file

4.4 Grid Testbeds

The resources used in our experiments consisted of both local resources at Melbourne and oversea resources located at Japan and Spain. The resources locally are Belle Grid Server at University of Melbourne which runs Globus toolkit 2.4 and ngdev which is a gateway machine running Globus toolkit 4 maintained by VPAC. There are 4 machines at University of Electronic and Communication Tokyo Japan which runs Globus Toolkit 4 and 2 clusters (a Torque cluster and a SGE cluster) and 2 machines located in Spain which serve the Globus Toolkit 4. The broker is designed to support management of job execution on remote resources using SSH-based connection and Globus-based access. For clusters running SGE or PBS, we established a SSH channel for staging the input data to a proper client node and ran the jobs submission command.

To define the resources which can be used by the Gridbus Broker, the user is required to give the information including the hostname and the platform

configurations (e.g. middleware setting, queue setting, firewall…). All these information is described using XGRL (XML-based Grid Resource Language) provided by the Gridbus Broker. Once the resources definition is given by XGRL, the broker is able to dispatch jobs at runtime to each of them depends on the availability and the scheduling algorithm.

5 Evaluation

After the first run of GSOM, a rough word category map is generated, which contains 35 nodes. The size of the data for each node ranges from 747kB to 6,041kB. These data are then bundled with the GSOM program and compressed to form the job tarball, which ranges from 468kB to 2,645kB. Figure 4 demonstrates the total jobs completed for each remote resource. Belle Grid Server (with 4 CPU) and Aquila SGE cluster server (with a queue limit 4) finished most of jobs, which was 12 and 7 respectively. And ngdev also finished 4 jobs without any queue. The execution time statistics for each job on the servers is shown in Figure 5. Some resources located in Japan and Spain performed worse compared with the resources at Melbourne, it was because that the network latency and broker overhead including stage-in and stage-out were much higher than the local resources. The time spent on staging in this experiment ranges from 2 to 20 seconds.

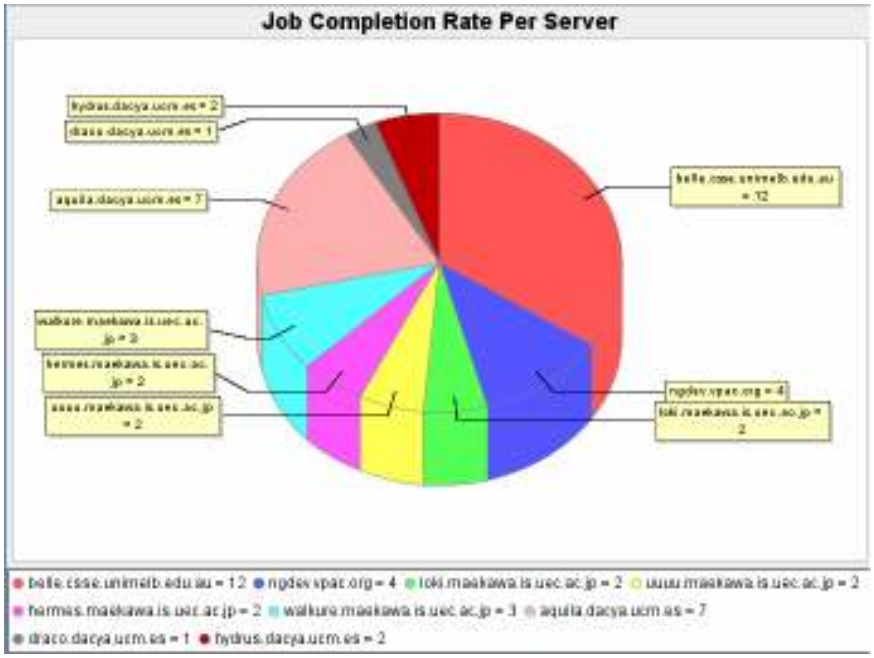

Figure 4. Jobs finished on each resource (generated by Gridbus Workbench).

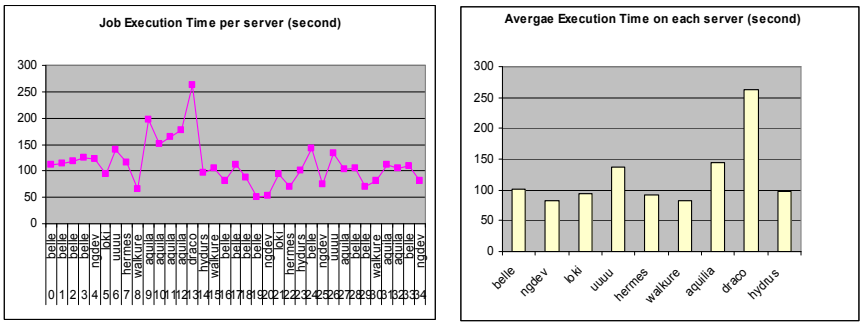

Figure 5. Job execution time statistics

Table 1 below summarizes the results of using a single GSOM, the simulation results from previous study and the ones from this experiment. Since the task of post processing can not be realized without the results from all the nodes. Therefore, the running time shown is the time taken to finish all the 35 jobs, plus the time spent in generating the rough word map.

Table 1. Execution time for generating the word map for Usenet articles

method	Total No. of nodes	Time (second)
Single node	402	5407
Simulation	752	347
Grid Testbeds	752	568

It is clear that the execution time obtain in this experiment is about 10 times less than the traditional way. The simulation takes the least time as expected, since we are assuming the ideal case of 35 Grid resources with no network latencies. Whereas we only use 9 resources in this experiment and there are other overheads such as the time spent in file transfer and polling time of the scheduler. It can be expected that the performance will improve further with more resources. Nonetheless, the scalable GSOM clustering method shows a very promising performance and is more suitable to process the huge amount of documents in large-scale applications.

6 Conclusion

In this paper, we have shown an implementation of a scalable GSOM text clustering method which takes the advantage of the Grid Computing technology. By splitting the workloads onto geographically distributed resources, the results indicate that there is a clear performance speedup in execution time on a computational grid compare to the traditional single-process approach. As a future development, it will be desirable to further integrating the grid broker within the clustering program. This can be achieved

by utilizing the APIs provided by the broker's library to invoke the services at runtime.

Acknowledgments. The work presented in this article was funded by Australian Research Council.

Reference

1. Honkela, T., Kaski, S., Lagus, K., Kohonen, T.: Newsgroup Exploration with WEBSOM Method and Browsing Interface, Tech. Rep. A32, Helsinki University of Technology, Laboratory of Computer and Information Science, Espoo, Finland (1996).
2. Kaski, S., Honkela, T., Lagus, K., Kohonen, T.: WEBSOM—Self-Organizing Maps of Document Collections, Neurocomputing 21 (1998) 101–117.
3. Foster, I., Kesselman, C. (eds.): The grid : blueprint for a new computing infrastructure, Amsterdam, Boston , Elsevier, 2004
4. Alahakoon. D., Halgamuge, S.K., Srinivasan, B.: Dynamic Self-Organising Maps with Controlled Growth for Knowledge Discovery, IEEE Transactions on Neural Networks, Special Issue on Knowledge Discovery and Data Mining, vol. 11, no. 3, 2000.
5. Lagus, K., Kaski, S., Kohonen, T.: Mining Massive Document Collections by the WEBSOM Method, Information Sciences, Vol 163/1-3, pp. 135-156, 2004.
6. Honkela, T.: Self-Organizing Maps in Natural Language Processing, Ph.D. thesis, Helsinki University of Technology, Neural Networks Research Center, Espoo, Finland, 1997.
7. Salton G.: Developments in Automatic Kohonen, T.: self-organizing maps, Springer-Verlag, Berlin, 1995
8. Hsu, A., Halgamuge, S.K.: Enhancement of Topology Preservation and Hierarchical Dynamic Self-Rrganising Maps for Data Visualisation, International Journal of Approximate Reasoning, vol. 32/2-3 pp. 259-279, Feb 2003.
9. Alahakoon, D.: Controlling the Spread of Dynamic Self Organising Maps, Neural Computing and Applications, 13(2), pp 168-174, Springer Verlag, 2004
10. Wickramasinghe, L.K., Alahakoon, L.D.: Dynamic Self Organizing Maps for Discovery and Sharing of Knowledge in Multi Agent Systems in Web Intelligence and Agent Systems: An International Journal, (IOS Press), Vol.3, No.1, 2005.
11. Zhai, YZ., Hsu, A., Halgamuge, S.K.: Scalable Dynamic Self-Organising Maps for Mining Massive Textual Data, 13th International Conference, ICONIP 2006, Hong Kong, China
12. Hughes, B., Bird, S., Lee, H., Klein, E., Experiments with data intensive NLP on a Computational Grid, Proceedings of the International Workshop on Human Language Technology, Hong Kong, 2004.
13. Hughes, B., Venugopal, S., Buyya, R., Grid-based Indexing of a Newswire Corpus, Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, IEEE Computer Society Press, Los Alnmitos, CA, USA, 2004
14. The Gridbus project, http://www.gridbus.org/