European Research Network on Foundations, Software Infrastructures and Applications for large scale distributed, GRID and Peer-to-Peer Technologies
A Network of Excellence funded by the European Commission

Multi-Cluster Text Mining on the Grid using the D-Grid UNICORE environment

Kai Kumpf, Theo Mevissen, Oliver Wäldrich, Wolfgang Ziegler
{kai.kumpf, theo.mevissen, oliver.waeldrich, wolfgang.ziegler}@scai.fraunhofer.de
Fraunhofer SCAI, Department of Bioinformatics
53754 Sankt Augustin, Germany

Sebastian Ginzel, Thomas Weuffel
Sebastian.Ginzel@smail.inf.fh-bonn-rhein-sieg.de
thomas@weuffel.de
University of Applied Sciences Bonn-Rhein-Sieg
53757 Sankt Augustin, Germany

Philipp Wieder
philipp.wieder@udo.edu
IT & Media Centre, University of Dortmund
44221 Dortmund, Germany

CoreGRID Technical Report
Number TR-0109
November 9, 2007

Institute on Resource Management and Scheduling

CoreGRID - Network of Excellence
URL: http://www.coregrid.net

CoreGRID is a Network of Excellence funded by the European Commission under the Sixth Framework Programme

Project no. FP6-004265

Multi-Cluster Text Mining on the Grid using the D-Grid UNICORE environment

Kai Kumpf, Theo Mevissen, Oliver Wäldrich, Wolfgang Ziegler
{kai.kumpf, theo.mevissen, oliver.waeldrich, wolfgang.ziegler}@scai.fraunhofer.de
Fraunhofer SCAI, Department of Bioinformatics
53754 Sankt Augustin, Germany

Sebastian Ginzel, Thomas Weuffel
Sebastian.Ginzel@smail.inf.fh-bonn-rhein-sieg.de
thomas@weuffel.de
University of Applied Sciences Bonn-Rhein-Sieg
53757 Sankt Augustin, Germany

Philipp Wieder
philipp.wieder@udo.edu
IT & Media Centre, University of Dortmund
44221 Dortmund, Germany

CoreGRID TR-0109
November 9, 2007

Abstract

Text mining is inherently more computation-intensive than information retrieval on pre-structured data, and it requires transfer and filtering of huge amounts of data. Grid environments provide a suitable infrastructure for accomplishing these tasks. We present the mapping and implementation of a standard text mining workflow for the analysis of biomedical text data from PubMed to a D-Grid UNICORE environment with multiple PC-clusters. We, furthermore, discuss the gain in applicability, the open issues of our solution and possible future enhancements.

1 Introduction

Parallel data mining strategies have long been established for different classes of data mining problems [23]. An important class of applications that lend themselves to parallelisation or task farming are long-running parallel computations with large numbers of independent tasks (Monte Carlo simulations, parameter-space searches, etc.) [2]. Text mining (TM) is one branch of data mining within which a considerable number of applications suitable for task-farming occur, especially for the pre-processing during which texts are structured using various tagging stages (part-of-speech tagging, named entity recognition).

Grid-enabled TM data interfaces and services provide mechanisms that allow users to identify (locate), access, integrate and interface distributed text sources and TM programs in a flexible way. This includes seamless access and selection of subsets of text corpora, data transfer, data pre-processing and meta-data functionality. TM services themselves are part of the vision of the Knowledge Grid [1]. The need for Grid-based TM solutions has also been

This research work is carried out under the FP6 Network of Excellence CoreGRID funded by the European Commission (Contract IST-2002-004265).

recognised by the Datamining Grid [4]. As demonstrated by IBM's Unstructured Information Management Architecture (UIMA) [8] approach, the analysis of unstructured data like text follows a simple pattern that can be expressed by a sequential or parallel series of TM filters. Optional splitting of input into appropriate chunks precedes the filtering stages, joining of partial results ends the workflow. A simple TM task like recognition and retrieval of keyword terms in all the individual documents of a domain specific text corpus (e.g. of biomedical PubMed abstracts) can be paralleled easily, because - in contrast to most challenges posed by parallel data mining - the paralleled processing stages can run independently without any communication overhead.

The remainder of the paper is organised as follows: In Section 2 we give an overview of related research, Section 3 contains a detailed description of the Grid environment, the text mining tools and the methodology used for implementing textmining in a UNICORE environment. In Section 4 and 5 we present and discuss the results of our experiments and Section 6 concludes the paper describing future work.

2 Related work

Certainly, the most outstanding examples of parallel text mining solutions exploiting task farming on clusters can be found in the indexing backends of Web crawlers that provide the basis for state-of-the-art Web search engines like Google. Since most of those are proprietary solutions, however, they cannot easily be adopted for general tasks in an open-standards environment. The implementation of TM algorithms to cluster computing using OpenMosix [21], a Linux kernel extension for single-system image clustering, was reported by Collier [3]. Ghanem et al. [10] describes an open TM workflow solution on the basis of Web services in Discovery Net [6]. Web service based frameworks for text mining were also proposed by Grover et al. [13] with a particular emphasis on workflow construction using Taverna [19] and Gaizaukas et al. [9]. Yu et al.[28] report the ongoing development of an Grid- and Web-based TM architecture but without enlarging on implementation details. To our knowledge, we present here the first working Grid-enabled solution based on open software that utilises multiple Grid clusters.

3 Environment, Tools, and Methods

In this section we present the basic characteristics of the Grid environment we were using, followed by a description of the text mining tools and the methodology we introduced to implement a distributed TM workflow.

D-Grid [5], the German national effort to create a sustainable infrastructure for e-Science, is co-funded by the Federal Ministry of Education and Research (BMBF) and the participating partners from academia, research centres and industry. D-Grid supports and provides distributions of three middleware systems, namely gLite [11], Globus Toolkit 4 [14], and UNICORE 5 [7]. These middlewares coexist within the D-Grid ecosystem to serve the needs of different application communities. A unified Grid infrastructure for D-Grid has never been targeted, higher-level services like scheduling or monitoring, however, which can operate across the different middlewares, are in the focus of D-Grid projects.

The resources of the VIOLA [26] testbed (see Figure 1), which we used to execute our experiments, have been made compliant to the D-Grid infrastructure requirements and have been made available to D-Grid users autumn 2006. As the VIOLA testbed is based on the UNICORE middleware, our TM implementation for Grid environments is set up on the UNICORE middleware, too. The TM framework, however, is by design middleware-agnostic and can be ported with reasonable effort to the other middleware systems supported by D-Grid.

3.1 Grid Environment

The UNICORE system provides seamless and secure access to distributed resources. Furthermore, UNICORE enables users to access compute systems with different hardware and software platforms in a uniform way, while these systems are potentially located in different organisational environments. Additionally, it supports user-side definition of high-level tasks, in particular workflow composition. The UNICORE version we use is not Web Services-compliant, since the Web Services-based UNICORE 6 was not available during the time we executed our experiment, but has been released mid of 2007 [18].

As shown in Figure 2 UNICORE introduces a three tier Grid architecture consisting of user, server and target system tier, an implementation of which is realised entirely in Java.

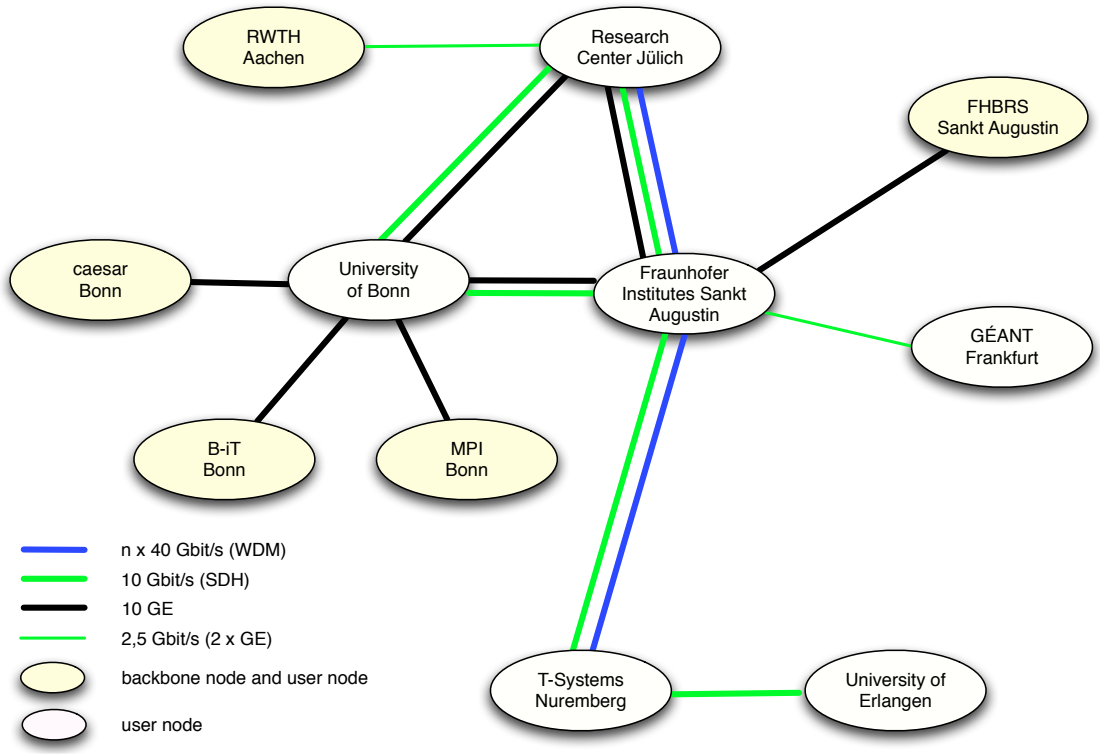

Figure 1: VIOLA (Vertically Integrated Optical testbed for Large Applications) Topology

The user tier is represented by the UNICORE Client which provides a graphical user interface to exploit the entire functionality offered by the server tier. This is achieved by sending and receiving Abstract Job Objects (AJO) via the UNICORE Protocol Layer (UPL). The AJO is the realisation of the job model and central to UNICORE's philosophy of abstraction and seamlessness. It contains platform and site neutral descriptions of computational and data related tasks, resource information and workflow specifications along with user and security information. AJOs are built within the user tier and sent formatted as serialised and signed Java objects to the Gateway.

The Gateway is the secure entry point to a UNICORE site, a Usite, which authenticates user requests and transfers them to the Network Job Supervisor (NJS). The NJS translates the abstract job into a target system specific one, a process called Incarnation, making use of the Incarnation Database (IDB). Furthermore the NJS contains a workflow engine which, among other tasks, performs pre- and post-staging of computational data and authorises the user by contacting the UNICORE User Database (UUDB). The Gateway and NJS execute typically on dedicated secure systems behind a firewall.

UNICORE's communication endpoint is the Target System Interface (TSI), a stateless daemon executing on the target system. It interfaces with the local resource manager represented either by a batch system like LoadLeveler, a batch system emulation on top of Linux (for example) or an entity capable of providing access to Grid resources like Globus.

3.2 Text mining Environment

The employed textmining software *Inside Discoverer* ^{TM}*extractor* (IDE) of TEMIS [25] was developed to extract domain specific information from arbitrary text sources. Specific knowledge resources, so-called skillCartridges, are plugged in to the IDE to perform domain specific analysis. The BER (Biological Entity Recognition) skillCartridge uses *ProMiner*[15],[16], an entity recognition system written in ANSI C, to identify gene and protein names as biological entities. The dictionary based ProMiner uses an approximate search algorithm extending the set of given names by different classes of spelling variants. The dictionary we used contains names of 32831 human genes and proteins with a total of 395065 different synonyms. IDE uses a grammar based approach to extract meaningful relations between

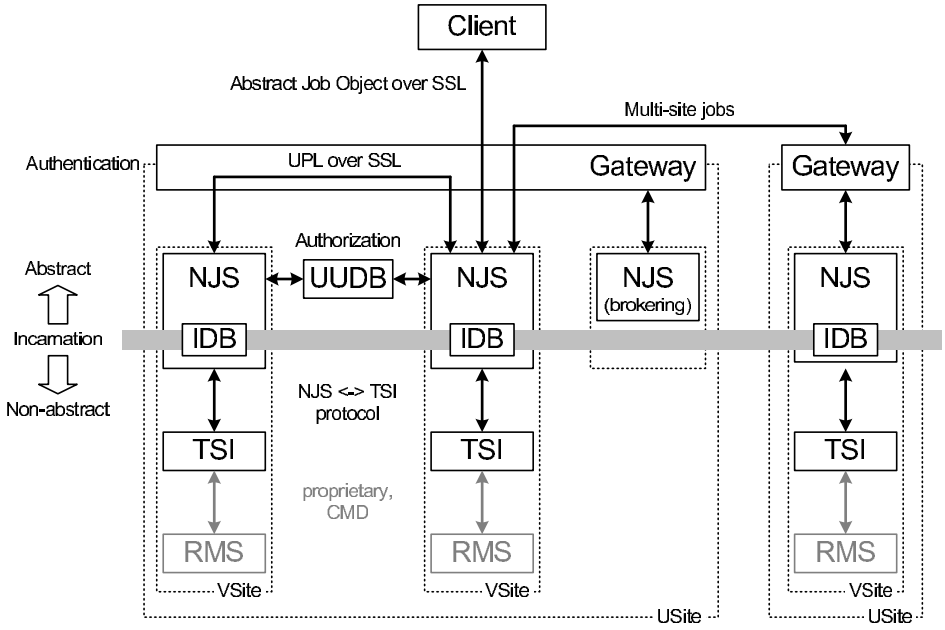

Figure 2: The UNICORE architecture

these entities from the text. It is implemented in Java.

In an initialisation phase, the IDE server has to be started. When the server is running, client processes can send pieces of text for analysis. The selected skillCartridge is started by the server, which launches the ProMiner for the entity recognition task. Reading the text pieces and writing the result file is handled by the client process. This is the "Document processing" stage within the Node process in figure 3.

In our environment a new server is started for every chunk of documents using one node exclusively. One client process running on the same node supports the server with the data. When all abstracts are scanned, the server and all related processes will be terminated.

3.3 Methodology

For the task at hand, we developed a shell-script wrapper solution using the UNICORE command line interface (CLI) developed in the OpenMolGRID project [20]. For management of the cluster resources we used the EASY scheduler [24] and PBS Professional [22]. As mentioned above, our TM filter consisted of a serial combination of the TeMiS tagger and Fraunhofer SCAI's ProMiner for biological entity recognition. The UNICORE system served as Grid middleware. The workflow was executed at three of the clusters of the VIOLA multi-cluster testbed depicted in figure 1.

The current production implementation of the standard TM workflow runs on one cluster and uses the local resource management system for submission of new jobs. All data is distributed and stored via the local cluster's NFS (network file system), which seems to cause sporadic unaccountable errors affecting the integrity of the resulting files. So far, each job runs an error detection and handling routine after the processing of the abstracts is finished. If an error is identified, the amount of documents is split in two parts and two new jobs are scheduled. At best this results in a job trying to process an invalid document. This invalid document is marked as erroneous and is removed from the further processing. For the sake of improved stability for data distribution and retrieval and because of the need for multi-cluster processing, an additional layer was added to this implementation of the workflow. The UNICORE middleware is used for job submission, status checking, and result fetching and operates on the top of the intra-cluster scheduler. An overview of the workflow is given in Figure 3. The hub host represents a known or discovered host in the Grid, that provides a common temporary file store. When using `scp` (secure copy) for file staging, this hub host is used by the nodes for retrieving the document packages and committing the result files.

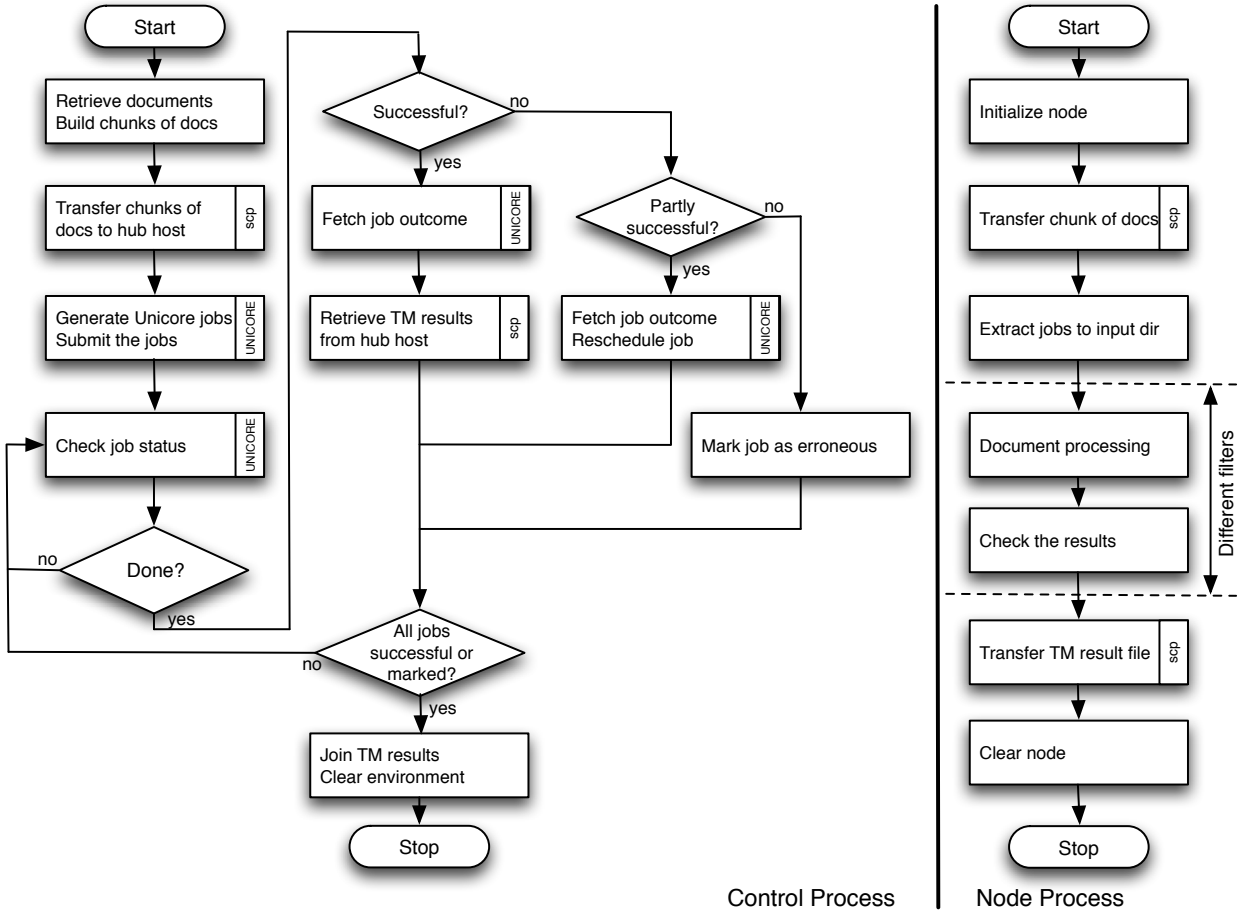

Figure 3: The TMGrid workflow. The actual processing stage on one Grid cluster node can be substituted by modules as appropriate. Limitations in UNICORE data staging force part of the workflow to be handled by the wrapper shell script.

4 Results

Processing of a corpus containing a defined amount of documents can be split into two parts.

The first entails retrieval of the corpus from a database, followed by splitting of the corpus to chunks of a defined constant number of documents. The transfer of these chunks to the hub host is followed by scheduling of the node process jobs and finally the retrieval of the results from the hub host. The duration of these pre-and post-processing stages depends on the total number of documents in the corpus and does not vary with the number of available nodes.

The intermediate part, document processing on individual cluster nodes, depends on the start-up and initialisation time of the TeMiS tagger and the ProMiner software, and the duration required for processing of the documents in a chunk.

The workflow described in Sections 3.2 and 3.3 was executed in the VIOLA multi-cluster testbed, using three PC clusters. The clusters are located at the Research Center Jülich, the University of Applied Sciences Bonn-Rhein-Sieg, and the Fraunhofer Institute SCAI. The clusters are interconnected with a 10 Gbit/s optical network and all nodes of the individual clusters are connected with 1 Gbit/s links to the local switches that are directly connected to the 10 Gbit/s backbone. Table 1 shows some experimental measurements, the numbers are plotted in Figure 4. The measurements are based on different runs with different combinations of clusters in the VIOLA testbed. The clusters have the following configurations:

1. PK (PACKCS): 14 nodes - 2x P-III processors and 2 Gigabyte RAM per node
2. BR (FH-BRS WR): 6 nodes - 4x Opteron processors and 8 Gigabyte RAM per node

3. CR (CRAY): 60 nodes - 2x Opteron processors per node and 100 Gigabyte shared RAM

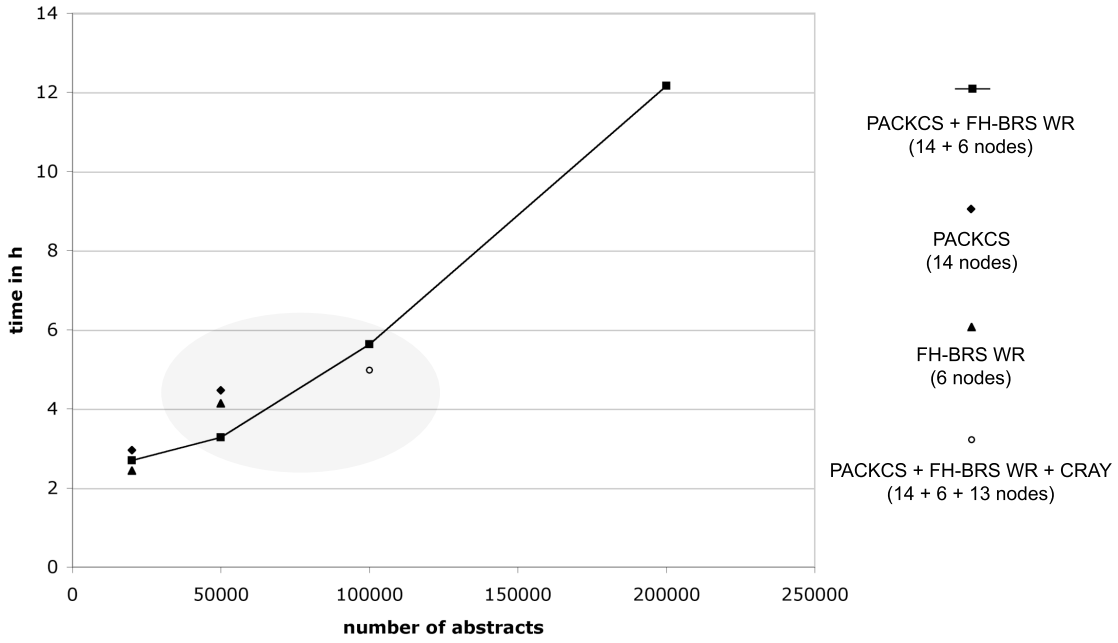

Figure 4: Results from table 1

Table 1: Experimental results for performance scaling in a multi cluster environment with varying numbers of nodes, clusters and documents. The result of measurement 5 is still higher in comparison with the result of 7. While 7 has to deal with 100.000 documents, it almost requires the same time as 5, in which 50.000 documents are processed.

	clusters	no.nodes.used	distribution	no.abstracts	pkg.size	time(hh:mm)
1	PK/BR	20	14 : 6	20.000	1.000	02 : 41
2	PK	14	–	20.000	1.000	02 : 57
3	BR	6	–	20.000	1.000	02 : 26
4	PK/BR	20	14 : 6	50.000	1.000	03 : 17
5	BR	6	–	50.000	2.500	04 : 08
6	PK	14	–	50.000	2.500	04 : 28
7	PK/BR	20	14 : 6	100.000	1.000	05 : 38
8	PK/BR/CR	33	14 : 6 : 13	100.000	1.000	04 : 59
9	PK/BR	20	14 : 6	200.000	2.500	12 : 09

The workflow itself scales sub-linearly. It is highly dependent on available clusters - or more precisely the number of available nodes - and the number of documents to process. For a small amount of documents, the number of available nodes is not that important, because the initialisation and start-up of the software packages on each node will limit the minimum processing time. For a large amount of documents, it is important to define an optimal chunk size of documents, since clusters with faster nodes could process more documents with the same initialisation overhead.

Based on the number of jobs and properties like the performance of each cluster's node, the current node workload and/or costs, it is important to favour clusters providing the desired properties and to allocate the jobs respectively.

5 Discussion

During the measurements two new aspects emerged, which will need further attention. The first issue is related to the number of submitted jobs. Since each one is created and submitted for every single chunk of documents, the Grid middleware has to handle quite a lot of jobs. As UNICORE has to track and trace the state of each job, the amount of jobs creates a considerable load on the Grid, in particular the UNICORE front-end server (Gateway). A better solution would be to create and schedule one job per cluster based on the current load. This job would reserve all required nodes for a specific time and execute a generic master script. Because the nodes are reserved, a script could start and control the processing of all documents destined for the cluster, in turn decreasing the load in the UNICORE infrastructure.

A second limitation occurred during the data distribution. UNICORE 5 uses its own Secure Sockets Layer (SSL)-based protocol which provides a reliable mechanism for transmission of all needed files. However, this data transfer mechanism has a quite limited transfer rate. Using the entire data stored in a database, the data transmission would already last several days. To circumvent this problem, we used `scp` as an alternative file staging mechanism. In production systems this approach is not applicable, since it bypasses the authentication and authorisation mechanisms provided by the Grid middleware. However, this approach was good enough for a proof of principle.

In the current case we are dealing with a special kind of TM workflow that can be implemented in an "embarrassingly parallel" way straightforwardly, because the processing of the individual data chunks does not require any communication between Grid nodes. The results presented here might change significantly when the goal of the workflow becomes overall indexing of a corpus using, e.g. TF-IDF [9], where summary statistics are part of the computation. Still, entity recognition in huge numbers of documents is an extremely relevant application in TM [17] and so we do see our architecture fit for mainstream applications.

6 Outlook

As already discussed in this paper, efficient approaches to distribute text mining applications to available systems in the D-Grid are needed. Users should simply submit a TM job to the Grid, and the infrastructure services (the Grid scheduler) distribute the load to the available compute systems according to the user requirements. This implies that these services must be able to discover (new) resources in the Grid, select suitable systems for the execution, transfer the input data to the selected systems, plan and execute the jobs, and finally collect the application results. Our goal is to extend the Metascheduling Service (MSS) [27] developed in the VIOLA project to provide these features for TM applications at the D-Grid.

To discover resources in the D-Grid environment standard UNICORE mechanisms can be used. Therefore, the MSS queries the D-Grid UNICORE gateway for all registered VSites. A VSite is a UNICORE virtual site, which represents a real computing system. In a second step the MSS determines which VSites are practically available for a user (authorisation filtering). This can be done by simply querying the resource properties of the discovered systems. Only systems that a user can access to will return a valid response.

In a next step, the MSS needs to select suitable systems for execution of the TM application. Since compute systems in a Grid environment are heterogeneous by nature, a mechanism for predicting the runtime of an TM application on the different target systems is required. Using statistical methods on historical data (past application runs) seems to be a promising approach to estimate the processing time of the application per unit of work, where a unit of work is one document or a set of documents. This data can be used by the MSS in conjunction with information on the current load of Grid systems to e.g. minimise the turnaround time of an application or to optimise the usage of the available compute resources.

Another major challenge is the distribution of the workload to the subsystems. The input files of the TM application must be staged into the target systems and the results must be collected after the job execution. Since TM applications usually deal with huge amounts of data (e.g. entire date stored in scientific databases) the time needed to distribute the data becomes a crucial factor for the total execution time of the application (turnaround time), from submission to results. To minimise this turnaround time on the one hand efficient file transfer protocols (e.g. GridFTP [12]) for data distribution must be used, and on the other hand the data transfers should be optimised. Furthermore, guarantees for network resources (e.g. the advance reservation of bandwidth), are an important factor. A Grid scheduling service can use such guarantees to assure that input data is available at the target systems at a specific time. This information can also be used for system selection, load distribution and resource reservation.

With the advent of Web Services-compliant UNICORE [18], it is expected that our system will combine the openness of previous approaches with the power of multi-cluster computing.

For a better structured approach to the node document processing sub-workflow we are currently evaluating the use of UIMA [8] which would allow more failure-robust plugging of TM filters than the file-based approach we showed here.

7 Summary

We present the architectural outline and implementation of a standard text mining workflow for analysis of biomedical text data to a UNICORE environment with multiple PC-clusters. The implementation is general enough to accommodate all kinds of unstructured data filters that can work in parallel without inter-node communication. This is the first implementation of text mining on a standard Grid middleware that clearly demonstrates the huge performance speedup potential in distributed computing.

8 Acknowledgement

The work reported in this paper is funded by the German Federal Ministry of Education and Research through the VIOLA project under grant #01AK605L. This paper also includes work carried out jointly within the CoreGRID Network of Excellence funded by the European Commission's IST programme under grant #004265.

References

[1] M. Cannataro and D. Talia. The knowledge grid. *Communications of the ACM, 46/1*, pages 89 – 93, 2003. http://grid.deis.unical.it/kgrid/.

[2] Henri Casanova, Myungho Kim, James S. Plank, and Jack J. Dongarra. Adaptive Scheduling for Task Farming with Grid Middleware. *Int. J. High Perform. Comput. Appl.*, 13(3):231–240, 1999.

[3] Nigel Collier. Evaluation of an Open Mosix Cluster for Text Mining. Poster on The 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2003), 2003.

[4] DataMiningGrid. Web site. 29 Mar 2007: <http://www.datamininggrid.org/>.

[5] D-Grid Initiative. Web site. 29 Mar 2007 <http://www.d-grid.de/>.

[6] Discovery on the Net. Web site. 29 Mar 2007: <http://www.discovery-on-the.net/>.

[7] D.Erwin (eds). UNICORE Plus Final Report. Technical report, Research Center Jülich, Germany, 2003. ISBN 3-00-011592-7.

[8] D. Ferrucci and A. Lally. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering, 10*, pages 327 – 348, 2004. Published online by Cambridge University Press 11 Oct 2004.

[9] Rob Gaizauskas, Neil Davis, George Demetriou, Yikun Guo, and Ian Roberts. Integrating Text Mining into Distributed Bioinformatics Workflows: A Web Services Implementation. In *SCC '04: Proceedings of the 2004 IEEE International Conference on Services Computing*, pages 145–152, Washington, DC, USA, 2004. IEEE Computer Society.

[10] Moustafa Ghanem, Yike Guo, Anthony Rowe, and Jonathan Ratcliffe. A Grid Infrastructure for Mixed Bioinformatics Data and Text Mining. In *3rd ACS/IEEE International Conference on Computer Systems and Application*, pages 41–45, January 2005.

[11] gLite - Ligthweight Middleware for Grid Computing . Web site. 14 Mar 2007 <http://glite.web.cern.ch/glite/>.

[12] GridFTP. Web site. 29 Mar 2007:
<http://www.globus.org/grid_software/data/gridftp.php>.

[13] Claire Grover, Harry Halpin, Ewan Klein, Jochen L. Leidner, Stephen Potter, Sebastian Riedel, Sally Scrutchin, and Richard Tobin. A Framework for Text Mining Services. In *In Proceedings of the UK e-Science Programme All Hands Meeting 2004 (AHM 2004)*, Nottingham, UK, 2004.

[14] Globus Toolkit 4. Web site. 14 Mar 2007 <http://www.globus.org/toolkit/>.

[15] D. Hanisch, J. Fluck, H.T. Mevissen, and R. Zimmer. Playing biology's name game: identifying protein names in scientific text. *Pacific Symposium on Biocomputing*, pages 403–14, 2003.

[16] D. Hanisch, K. Fundel, J. Fluck, H.T. Mevissen, and R. Zimmer. Prominer: rule-based protein and gene entity recognition. *BMC Bioinformatics 2005*, 6:14v, 2005.

[17] Ulf Leser and Jrg Hakenberg. What makes a gene name? named entity recognition in the biomedical literature. *Brief Bioinform*, 6(4):357–369, Dec 2005.

[18] R. Menday. The Web Services Architecture and the UNICORE Gateway. In *AICT-ICIW '06. Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet*, page 134. IEEE Computer Society, February 19–25, 2006.

[19] Tom Oinn, Matthew Addis, Justin Ferris, Darren Marvin, Martin Senger, Mark Greenwood, Tim Carver, Kevin Glover, Matthew R Pocock, Anil Wipat, and Peter Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, Nov 2004.

[20] Openmolgrid. WWW. http://www.openmolgrid.org/.

[21] OpenMosix. Web site. 29 Mar 2007: <http://openmosix.sourceforge.net/>.

[22] PBS Pro. Web site. 19 Mar 2007 <http://www.altair.com/software/pbspro.htm>.

[23] David Skillicorn. Strategies for Parallel Data Mining. *IEEE Concurrency*, 7(4):26–35, 1999.

[24] J. Skovira, W. Chan, H. Zhou, and D. Lifka. The EASY — LoadLeveler API Project. In D. G. Feitelson and L. Rudolph, editors, *Proc. of 2nd Workshop on Job Scheduling Strategies for Parallel Processing*, volume 1162 of *Lecture Notes in Computer Science*, pages 41–47. Springer, 1996.

[25] Temis. Web site. 29 Mar 2007: <http://www.temis-group.com/>.

[26] VIOLA – Vertically Integrated Optical Testbed for Large Application in DFN, 2006. Web site. 29 Mar 2006: <http://www.viola-testbed.de/>.

[27] O. Wäldrich, Ph. Wieder, and W. Ziegler. A Meta-scheduling Service for Co-allocating Arbitrary Types of Resources. In R. Wyrzykowski, J. Dongarra, N. Meyer, and J. Wasniewski, editors, *Proc. of the Sixth International Conference on Parallel Processing and Applied Mathematics (PPAM 2005)*, volume 3911 of *LNCS*, pages 782–791. Springer, 2006.

[28] Lean Yu and Shouyang Wang Kin Keung Lai Yue Wu. A Framework of Web-based Text Mining on the Grid. In *Proceedings of the International Conference on Next Generation Web Services Practices (NWeSP'05)*. IEEE Computer Soc., 2005.