Grid-based Indexing of a Newswire Corpus

Baden Hughes, Srikumar Venugopal and Rajkumar Buyya
Department of Computer Science and Software Engineering
The University of Melbourne, Australia
{badenh, srikumar, raj}@cs.mu.oz.au

Abstract

In this paper we report experience in the use of computational grids in the domain of natural language processing, particularly in the area of information extraction, to create query indices for information retrieval tasks. Given the prevalence of large corpora in the natural language processing domain, computational grids offer significant utility to researchers in the domain who are reaching the bounds of computational efficiency. We leverage the affinities between the segmented data sources prevalent in natural language processing and the parallelisation model from the grid domain. The experiment reported here is a large-scale newswire corpus indexing task, with the goal to efficiently create a queryable index of the entire corpus. By parallelising the indexing task and executing it on an Australian computational grid, we observe overall performance improvement of a 2.26x speedup over the same experiment on a single computational node. In addition to reporting the raw performance impact, we reflect on a number of interesting points discovered during the execution of the experiments and propose a number of new requirements for grid middleware.

1 Introduction

Grid computing [9] enables the sharing and aggregation of geographically distributed high-end computers, networks and databases for solving large-scale problems in science, engineering and commerce. By enabling access to computational resources, data collections and remote instrumentation, grid computing has encouraged symbiosis between various complimentary domains in the pursuit of common goals. The potential for collaboration has sparked interest in grid computing within diverse areas such as high-energy physics, molecular biology and natural language processing. Grids are therefore, emerging as the computing platform of choice for the next generation of large-scale scientific research, also sometimes called eScience [21].

The natural language processing (NLP) domain in focus here is a broad discipline. One common feature across diverse NLP research is the use of large, collated corpora, or collections of naturally-occurring human language in written and/or spoken forms. In many areas, NLP has a long history of computational innovation; and increasingly, is reaching the bounds of computational efficiency. Thus NLP researchers are beginning to seek new approaches to large-scale analysis of human language data.

In the experiment described here, we process a text-based broadcast news media corpora, and derive a queryable index for information retrieval tasks. Newswire is a common data type, provided by large news agencies, and designed for a content syndication by smaller publishers. Newswire content is essentially a continuous stream of text with little internal structure, inherited from earlier implementations where telegraph and serial line printers were used to receive the data. In electronic form, the majority of newswire services are provided in basic Standard Generalized Markup Language (SGML).

Newswire is internally indexed by the use of series of coded tags called ‘slugs’. However, owing to its stream-based nature, external indices such as those required for efficient querying are not available by default. For research on the content of newswire corpora, such indices are required to be manually constructed. The complexity of such an indexing task increases exponentially with the increased volume of newswire sources. An efficient indexing approach would filter the newswire sources by content type, extract the most pertinent key phrases for each story, and construct an inverted index which references phrases against locations in the raw newswire corpus.

Since the newswire corpus is naturally segmented, parallelisation of the indexing process is a convenient approach to adopt. Here we use the natural divisions in the newswire corpus as the basis for parallelisation; distributing and executing the index creation application across a computational grid, and aggregating the results into a single master index. We show comparative results for the same process on a single computational node and on an Australian computational

grid, and find there are, as to be expected, significant performance gains from this mode of exection.

The remainder of this paper is organized as follows: we commence by providing a brief overview of the motivations for approaching computational grids from within natural language engineering application domain. We consider related work in both the grid and natural language processing communities. Next we describe the architecture, data set, application, grid interface, and grid infrastructure used in the experiment. Our results are reported and evaluated, followed by some general observations about the domain specific features which impact on the efficiency of widely-adopted computational grid models. Finally, we describe a number of directions for future work.

2 Motivation

The motivation for NLP researchers to adopt computational models such as distributed, grid and cluster computing may not appear obvious at first glance. However, even a cursory review of typical computational approaches prevalent within NLP indicate a strong affinity for such techniques.

NLP applications are typically constructed from a number of processing components, each responsible for a specialized task. Typical components include speech recognition, tagging, entity detection, anaphora resolution, parsing, etc. Each component is heavily parameterized and must be trained on very large datasets. Discovering optimal parameterizations is both data- and processor-intensive. Building complex applications, such as spoken dialogue systems, depends on identifying and integrating suitable components often from a range of sources.

In addition to heavy parameterisation, NLP also makes significant use of very large collections of human language data. Text corpora of more than a billion words are increasingly common, and multimodal corpora of thousands of hours of speech are being actively curated. From a data processing perspective alone, NLP requires high performance computational facilities to approach acceptable throughput.

It is now not uncommon to find references to NLP research which is unable to be completed owing to the bounds of the computational efficiency of traditional serialised processing. An excellent example is provided by Salomon et. al. [15] who posit that for the exhaustive combinatorical traversal of the multi-dimensional space represented by speech phones in a corpus, an estimated 6 years of CPU time would be required to complete the task. This constraint is not uncommon in NLP as a whole.

NLP research is increasingly empirical in orientation, and with such a data-intensive approach, the need for computational gains is reinforced. Many algorithms within NLP have not been tested for scaling over large data sets; nor have the bounds of many algorithms been quantified. As such research becomes more pressing, so to does the need for new models of computation which allow collaborative, distributed, data-intensive research.

Hence the motivation for NLP researchers to consider the grid computing paradigm is obvious. The natural affinity between segmented corpora prevalent in NLP and the parallel execution mode supported by the grid model are beginning to be exploited. The potential for grid-enabled parameter sweep applications to enable common tasks such as language modelling [14] and word-frequency counting [14] is beginning to be recognised within the discipline. The field is seeking new enablement models for data-intensive experiments which both allow existing techniques to be evaluated as to their accuracy and efficiency, and for new techniques to be developed to reveal new aspects of the system of human language.

3 Related Work

The work reported here is derivative of two distinct streams of research, one within the grid computing domain, and the other within NLP.

Grid and web service technologies have been applied successfully in many scientific and business domains. Some examples are high energy physics [3], astronomy [24] protein aggregation simulation [23] and medical image processing [16]. Grid resource brokers provide an abstraction from the complexity of grids by undertaking tasks for resource discovery, job scheduling, execution and monitoring and job output retrieval among others. Nimrod-G [4] is such a Grid resource broker which follows the parameter-sweep model of Grid execution, i.e., executing an application over a combinatorial range of values in its parameter space. It has been used in grid-enabling scientific applications such as modelling of chemical processes [17], drug discovery [5] and brain activity analysis [6]. The Gridbus broker [22] extends this well-known model to execute distributed data-intensive applications. As will be shown later, the Gridbus broker uses a native XML format for input which would be invaluable for defining future NLP-specific schemas. Here we use the Gridbus broker to grid-enable the newswire indexing application.

NLP shares some common characteristics with the aforementioned applications - namely large collections of data and the requirement for huge amounts of computational power to process this data. However, most of the above applications are numerically-oriented and the size of the data has a direct relationship with the amount of computing power required. Within NLP, as we have noted in our observations, this may not be the case.

The benefits offered by computational grids are slowly being realised within the NLP community. A review of

a number of recent contributions follows. Curran [7] provided an overview of the architecture of a high performance computation environment for NLP generally, and specifically for language learning tasks. Hughes and Bird [13] proposed a closer integration between natural language processing approaches and the broker-mediated computational grid environment. Furthermore, Hughes and Bird [12] refined and proposed an XML-grounded, workflow oriented upper middleware layer which provides ease of interface with grid brokering services. More recently, Tamburini [18] provided further evidence as to the utility of computational grids for NLP as a method for distributed corpora collation. Finally, empirical experiments have been conducted by Hughes et al [14] showing the significant benefit of computational grids in data-intensive natural language processing.

In both the computational grid and NLP domains the convergent themes of research are that where the bounds of computational efficiency have been met or exceeded, ease of acccess to computational resources can facilitate new types of analysis. Additionally we propose that the natural affinity of segmented large corpora resources prevalent in natural language processing with the data-oriented approach to grid-based computation offer a compelling solution which will enable cooperative, data-intensive, natural language processing research.

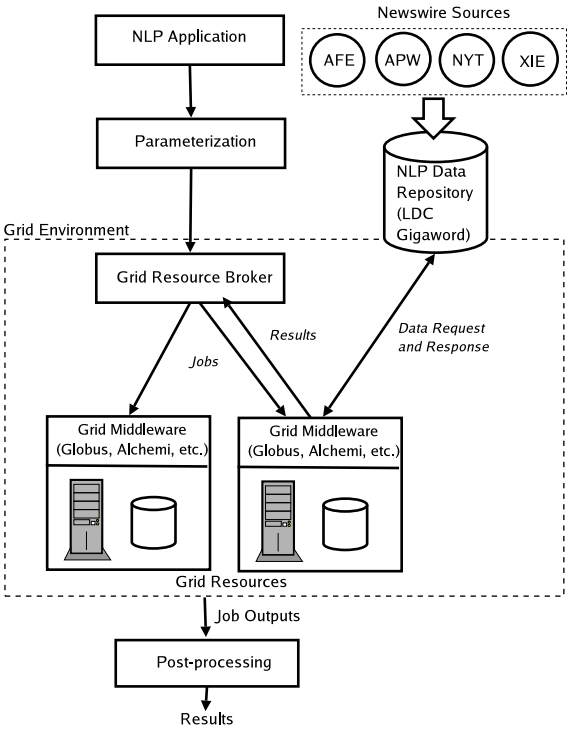

Figure 1. High-Level Architecture

4 Architecture

Figure 1 shows the high-level architecture that has been used in conducting this experiment.

The newswire corpora are archived and are stored in a repository. This repository is then made available to the grid through standard data transfer protocols such as GSIFTP. The natural language processing application is decomposed as a parameter-sweep application using the natural corpus segmentation. The parameterisation is then encoded as input to the grid resource broker. The broker discovers available computational resources, creates a job schedule and dispatches jobs to remote resources for execution. The broker monitors the dispatched jobs and retrieves their output after completion. At the end of the execution phase, the output of all the jobs is collated and post-processed to obtain the final results.

The next four subsections consider each of these phases in greater detail.

4.1 NLP Corpora

Typically a newswire service is a dedicated feed of stories from a larger news agency such as Reuters or AAP, which is provided to smaller content aggregators for syndication. Newswire content is primarily received via a dedicated subscription circuit such as a leased line or satellite service.

The corpus used in this experiment is the English Gigaword Corpus [10] from the Linguistic Data Consortium. This corpus is a collection of four international sources of English newswire, from Agence France Press English Service (afe), Associated Press Worldstream English Service (apw), The New York Times Newswire Service (nyt), and The Xinhua News Agency English Service (xie).

LDC English Gigaword Corpus

Source	Files	Mb-gzip	Mb	M-words	M-docs
afe	44	417	1,216	171	.656
apw	91	1,213	3,647	540	1.48
nyt	96	2,104	5,906	914	1.30
xie	83	320	940	132	.679
TOTAL	**314**	**4,054**	**11,709**	**1,757**	**4.11**

Newswire content is essentially a continuous stream of text with little internal structure. In electronic form, the majority of newswire services are provided in basic SGML. Newswire content is relatively error free, but may contain occasional transmission or human errors. Newswire services themselves vary between stream-based and chunk-based delivery modes - realtime stream-based services don't have much structural differentiation, whilst chunk-based services have better grouping of content and delineation between content types.

Within newswire content there is inherent duplication - transmissions are repeated, sometimes with minor alterations as stories change, or sometimes exact copies, depending on the delivery mode and editorial process of the source agency. Furthermore, the style of the newswire may vary based on orientation of the supplier (eg official government information sources vs independent commercial news operations). Another complication is that in some cases newswire services are multilingual, providing multiple translations of each story, thus interleaving linguistic variation along with structural complexity.

4.2 Indexing Application

The overall purpose of the application is to create a query index for the newswire corpus, which will allow efficient key phrase search and retrieval operations to identify sections of interest within the overall corpus. Given that newswire is natively a stream-based data source with only incidental internal indexing, the creation of such a reference index can only be achieved by processing the individual corpus segments and aggregating the results into a master index. Thus, the task of the indexing application itself is to construct extract the most pertinent key phrases for each story, and construct an inverted index which references two-word sequences against locations in the raw newswire corpus. The index itself can then be queried by external parties, and provides Key-Word-in-Context-style output together with document identifier and the headline from each story relevant to the query.

While newswire content is divided into 4 main categories (story, multi, other, advis), it is only the story category (which contains actual narrative content) which is in focus here. The other categories are significantly artificial in their construction and are not designed for average human consumption, but are rather codifications for news editors.

For the purposes of grid-enabling the application, the indexing application has been decomposed into three distinct parts from the original serialised version. Certain modifications were inevitable given the requirement that the application be dynamically parameterised at run time rather than making a linear pass through the data, however the indexing task is essentially the same.

4.3 Gridbus Broker and Parameterisation

The Gridbus broker is able to make scheduling decisions on where to place the jobs on the Grid depending on the computational resources characteristics (such as availability, capability, and cost), the user's quality-of-service requirements such as the deadline and budget, and the proximity of the required data or its replicas to the computational resources. However, for this experiment, we have conducted studies using a simple adaptive scheduling policy with load balancing, as the experiment output is simply an aggregate of all jobs, and all the source data is distributed from the central node at runtime.

The entire indexing application consists of 3 stages. The first part, executed on the broker host, derives the execution parameters for a given corpus segment, conducting a linear pass through all segments of the corpus, and creates a job tarball containing the indexing script supplemented by the parameters dynamically determined from the sweep. The second part, which is distributed, grid-enabled and managed by the broker, involves the execution of the indexing application on each corpus segment. The job tarball is uncompressed, the indexing script executed on the corpus segment, results obtained and transferred to the broker host, and cleanup performed. The third part, executed on the broker host, collates the output of the individual corpus segments, and aggregates them into two indices - one for each corpus, and one for the whole corpus.

A sample plan file for the execution stage, modelled after Nimrod-G's declarative programming language [1], and expressed in the native XML format accepted by the Gridbus broker is shown in Figure 2. The division of the corpora into archives based on months and years lends itself easily to parameterisation. The *arch* parameter selects the corpus on which the indexing is to be done. This parameter can be varied to select a single corpus or multiple corpora for simultaneous analysis. The other parameters are self-explanatory. By default, the parameterization process creates job objects within the broker based on the cross-product of all the values of all defined parameters. This causes problems for analysis of discontinous corpora in which the newswire archives of certain months or years may not be present, thus needlessly causing the failure of jobs based on the missing archives. To avoid this condition, in the current experiments, we were forced to override the plan file analysis by creating the parameter collections outside the broker and providing these as the input for cross-production. In future work, we plan to restrict the parameter value combinations by introducing user-defined validation conditions for parameter value sets.

4.4 Grid Infrastructure

Figure 3 shows the Grid testbed used in our experiments. We have deployed the application on a subset of the resources that are part of the Belle Analysis Data Grid [2] testbed, setup in collaboration with IBM, and a cluster at the Victorian Partnership for Advanced Computing (VPAC). The nodes were connected by GrangeNet (Grid and Next Generation Network) [11], a multi-gigabit network supporting grid and advanced communications services between academic institutions across Australia. The broker was de-

```
  <parameter>
     <name> year</name>
     <domain> <range> <from> 1994</from> <to> 2002</to>  <step> 1 </step> </range> </domain>
  </parameter>
  <parameter>
     <name> month</name>
     <domain><range><from> 1</from><to>12</to><step>1</step></range></domain>
  </parameter>
  <parameter>
      <name>arch</name>
      <domain>
        <select_anyof> <text>
           <value_list> afe apw nyt xie</value_list>
           <default>afe apw nyt xie</default></text>
         </select_anyof></domain>
  </parameter>
  <task>
      <type> main</type>
      <copy>
         <source> <location> <nospec/> </location> <file>$arch/jobs/$arch$year$month.job.tar.gz</file> </source>
         <destination><location><node/></location><file> $arch$year$month.job.tar.gz</file></destination>
      </copy>
      <execute>
        <location> <node/> </location>
        <command> gunzip $arch$year$month.job.tar.gz</command>
      </execute>
      <execute>
        <location><node/></location>
        <command> tar -xvf $arch$year$month.job.tar</command>
       </execute>
       <execute>
         <location><node/></location>
         <command>./$arch$year$month.index.sh</command>
        </execute>
        <copy>
           <source><location><node/></location><file>$arch-out.docids+head</file></source>
           <destination><location><nospec/></location><file>output/$arch-out.docids+head.$month.$year</file></destina
tion>
        </copy>
    </task>
 </plan>
```

Figure 2. Parameter-sweep specification file for NLP task

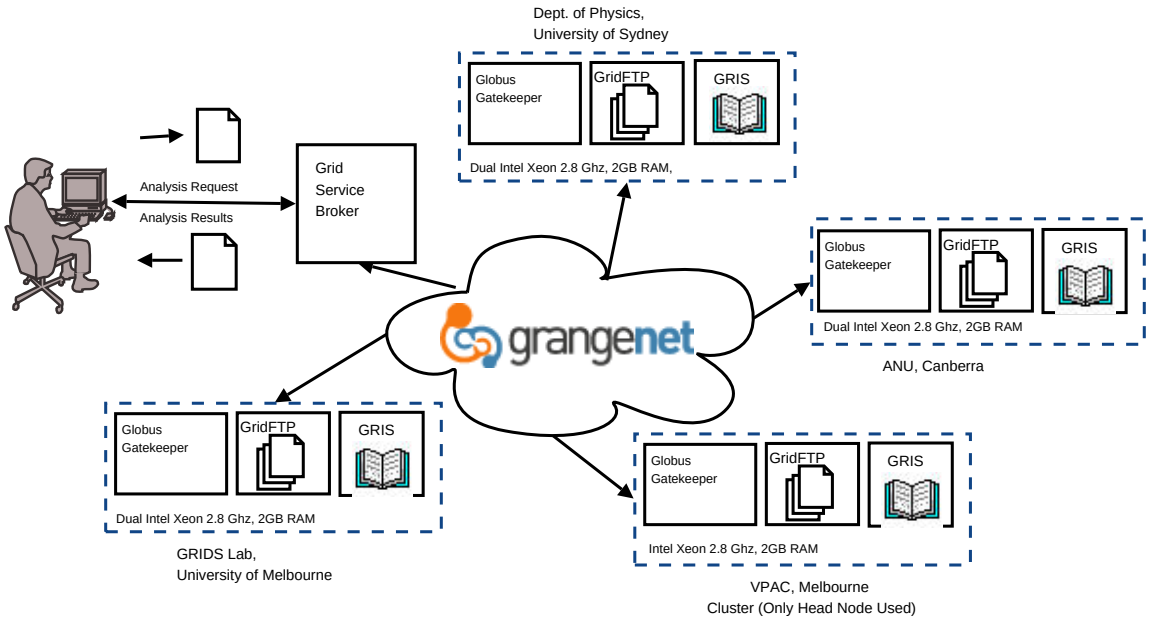

Figure 3. The testbed used for NLP experiments

ployed on the Melbourne GRIDS Lab machine from where jobs were dispatched to the other nodes at runtime.

All the nodes were running Globus 2.4.2 [8] which provided services for job submission and monitoring. The file transfer was done through GSIFTP and Globus GASS.

5 Experimental Results

It can be seen from the results below that a significant performance improvement was obtained by executing this task in parallel on an Australian computational grid described in the previous section, compared to execution in serial on a single node.

First we consider the elapsed time taken for the two different modes of computation as can be seen in the tables below.

Elapsed Time (Seconds) - Centralised Mode

	Job.Gen	Job.Exec	Job.Collate	Total
afe	79	1,863	18	1,960
apw	330	3,777	44	4,151
nyt	576	4,003	42	4,621
xie	62	3,407	20	3,489
Total	**1,047**	**13,050**	**144**	**14,221**

Elapsed Time (Seconds) - Distributed Mode

	Job.Gen	Job.Exec	Job.Collate	Total
afe	79	834	18	931
apw	330	1,438	44	1,812
nyt	576	1,907	42	2,525
xie	62	937	20	1,019
Total	**1,047**	**5,116**	**144**	**6,287**

In the figures above, it should be observed that only certain parts of the application were actually executed on the computational grid, namely the indexing phase. The pre- and post- processing parts were executed in serial in both the centralised and distributed modes, owing to their proportionally smaller computational requirements.

Comparing results for the elapsed time metric, the performance improvement on a per corpus segment basis ranged from 1:2.10x for afe, 1:2.29 for apw, 1:1.83x for nyt, 1:3.42 for xie; with an average per segment improvement of 1:2.41x. Overall we report a whole corpus performance improvement of 1:2.26x.

Next we consider a derived metric, words per second, for the two different modes of computation, in the tables below.

Words Processed per Second - Centralised Mode

	Words	Time (Sec)	Words/Sec
afe	170,969,000	1,960	87,229
apw	539,665,000	4,151	130,008
nyt	914,159,000	4,621	197,827
xie	131,711,000	3,489	37,750
Total	**1,756,504,000**	**14,221**	**123,514**

Words Processed per Second - Distributed Mode

	Words	Time (Sec)	Words/Sec
afe	170,969,000	931	183,640
apw	539,665,000	1,812	297,828
nyt	914,159,000	2,525	362,043
xie	131,711,000	1,019	129,255
Total	**1,756,504,000**	**5,116**	**343,335**

Comparing results for the words per second metric, the performance improvement on a per corpus segment basis ranged from 1:2.10x for afe, 1:2.29 for apw, 1:1.83x for nyt, 1:3.42 for xie; with an average per segment improvement of 1:2.41x. Overall we report a whole corpus performance improvement of 1:2.26x.

In both the elapsed time metric and the words per second metric we observe a performance improvement which is significantly less than linear when compared to the number of CPUs used for execution. While we do see an improvement in overall performance, the comparative ratios are 1:2.26 for elapsed time and/or words per second, as opposed to 1:3.5 for the number of CPUs utilised. In part this indicates that the task itself is not CPU bound so much as I/O bound. Another important point here is that in distributed mode, the bandwidth latency factor is much higher and is derived from the average between the broker and all nodes rather than between the broker and any single node. However, the raw performance improvement is significant enough to warrant further optimisation (eg adaptive scheduling) of the application instance. An improvement of 2.26x may represent in aggregate application contexts a significant performance gain and should not be discounted simply because it is not linear.

Our other observations based on these results are fourfold. First, if we derive a per node average of the words per second metric for the 4-node grid execution instance, we can see that the average throughput per node is significantly less in distributed mode (85,833 words per second) than in centralised mode (123,514 words per second). While it is not immediately clear as to the origin of this performance degradation, we can posit two viable reasons for it: 1) the non-symmetrical nature of the nodes on the computational grid in terms of the number of CPUs and 2) the impact of overlapping jobs as dispatched by the broker. Second, there is a significant variation in the words per second metric between corpus segments. This can be accounted for by the previously mentioned variable distribution of the items of interest within a corpus, namely, those of type story in relation to other types.

Third we can consider the comparative job execution rate for the distributed vs centralized segment of the indexing task (since that is the only point which differs between the two experiments). We can observe from that overall the execution time is much shorter under distributed mode when compared to centralized mode. This is to be expected, as

Figure 4. Comparative Job Completion

in distributed mode there are a greater number of jobs completed per unit time based on parallelisation.

Fourth, and finally, from Figure 5 we can see that the Gridbus broker distributed the individual jobs relatively evenly across the lifetime of the experiment. One anomaly is that the host brecca2.vpac.org only received a relatively small number of jobs in the afe corpus analysis. We can attribute this anomaly to two factors: 1) the host itself has only a single CPU, rather than a dual CPU architecture, and thus broker-detected metrics as to the overall load on the machine would be impacted; and 2) the bandwidth between the broker host and this particular computational node was in fact the most variable.

6 Observations

A number of interesting general observations can now be made.

In common with the findings in a number of other domains, not all tasks in natural language processing are ripe for parallelisation. We can see from the results that although not all aspects of the application are grid-enabled, the time elapsed in pre- and post- processing is insignificant compared to the time taken to actually index the corpus segments. Hence there would be little benefit gained from distributing these parts of the overall application - in fact the time required to distribute them would surpass the computation time thus rendering such efforts inherently inefficient.

We also note that the natural segmentation of corpora is an important factor in reviewing performance gains - it appears that a greater proportional speedup is gained in corpora which have fewer but larger files (eg xie, with 1:3.42), than in those which have more but smaller files (eg nyt, with 1:1.83. This is possibly counter-intutive to a fundamental tenet of parallel computing, that a greater degree of parallelisation should result in greater performance.

Furthermore, we note that the typology of corpora content is an important factor. Since we only process items of type `Story`, then the relative number of these items in relation to all other items in a corpus segment can impact performance, within each segment we are essentially conducting a linear pass to find such data instances.

It is also interesting to note that much of the processing within this experiment required transfer of the segmented archives from the broker host to the compute resources. It is possible to gain improved performance by minimising the data transfer time, although in this particular case, a high bandwidth connection between all sites does mean that a relatively small amount of optimisation could be gained in this fashion. In the immediate future, we would like to apply data-oriented scheduling, i.e., scheduling taking into account bandwidth considerations and the relative size of each job, into this domain and contrast its performance with the computational scheduling to measure the benefits gained and to establish the minimum I/O:computation ratio required for any efficiency gain.

7 Conclusion and Future Work

In this paper we have shown the adaptation of a NLP application, an indexer for newswire sources, to be executed on a computational grid. The input data source is naturally segmented, allowing the application to be easily parameterized, and thus deployed. It can be seen from the results reported that there is a clear performance benefit in executing this NLP application on a computational grid.

However, much work is required before grid solutions can be applied within the NLP domain on a daily basis by ordinary researchers. A large number of NLP applications can be construed as only requiring grid-based analysis at a single point in the overall workflow. Thus, there is a need for integration of services provided by grid brokers within common NLP application frameworks. We note that already this work has begun in the context of the Annotation Graph Toolkit [19] and the Natural Language Toolkit [20].

References

[1] D. Abramson, R. Sosic, J. Giddy, and B. Hall. Nimrod: A tool for performing parameterised simulations using distributed workstations. In *Proceedings of the 4th IEEE Symposium on High Performance Distributed Computing (HPDC)*. IEEE Computer Soceity Press, USA, 1995.

[2] Australian Belle Analysis Data Grid . http://roberts.ph.unimelb.edu.au/epp/grid/badg/. Accessed June 2004.

[3] J. Bunn and H. Newman. *Data-intensive grids for high-energy physics*. John Wiley and Sons, Inc., New York, 2003.

[4] R. Buyya, D. Abramson, and J. Giddy. Nimrod/G: An Architecture for a Resource Management and Scheduling System in a Global Computational Grid. In *Proceedings of the*

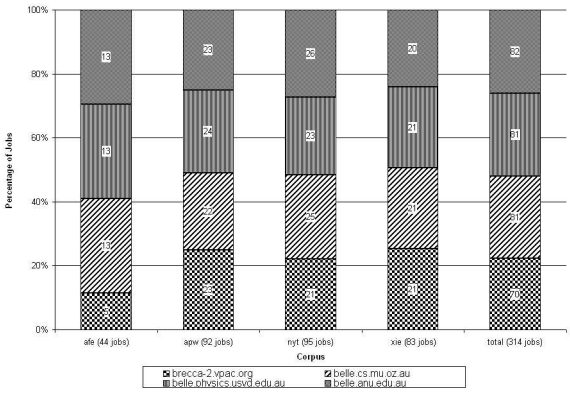

Figure 5. Job Distribution

4th International Conference on High Performance Computing in Asia-Pacifi c Region (HPC Asia 2000), Beijing, China, 2000. IEEE Computer Press, USA.

[5] R. Buyya, K. Branson, J. Giddy, and D. Abramson. The virtual laboratory: Enabling molecular modeling for drug design on the world wide grid. *The Journal of Concurrency and Computation: Practice and Experience*, 15:1–25, 2003.

[6] R. Buyya, S. Date, Y. Mizuno-Matsumoto, S. Venugopal, and D. Abramson. Neuroscience Instrumentation and Distributed Analysis of Brain Activity Data: A Case for eScience on Global Grids. *Journal of Concurrency and Computation: Practice and Experience*. Accepted and in print.

[7] J. Curran. Blueprint for a High Performance NLP Infrastructure. In *HLT-NAACL 2003 Workshop on Software Engineering and Architecture of Language Technology Systems*, pages 40–45. Association for Computational Linguistics, 2003.

[8] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11:115–128, 1997.

[9] I. Foster and C. Kesselman. *The Grid: Blueprint for a Future Computing Infrastructure*. Morgan Kaufmann Publishers, 1999.

[10] D. Graff. *English Gigaword*. Linguistic Data Consortium, Philadelphia, 2003. LDC Catalog LDC2003T05 ISBN 1-58563-260-0.

[11] GrangeNet (GRid And Next GEneration Network). http://www.grangenet.net. Accessed Jun 2004.

[12] B. Hughes and S. Bird. A Grid Based Architecture for High Performance NLP. *Paper submitted to Natural Language Engineering*, 2003.

[13] B. Hughes and S. Bird. Grid-Enabling Natural Language Engineering By Stealth. In *HLT-NAACL 2003 Workshop on Software Engineering and Architecture of Language Technology Systems*, pages 31–38. Association for Computational Linguistics, 2003.

[14] B. Hughes, S. Bird, H. Lee, and E. Klein. Experiments with Data-Intensive NLP on a Computational Grid. In *Proceedings of the International Workshop on Human Language Technology*. University of Hong Kong, 2004.

[15] J. Salomon, S. King, and M. Osborne. Framewise phone classifi cation using support vector machines. In *7th International Conference on Spoken Language Processing*, 2002.

[16] S. Smallen, H. Casanova, and F. Berman. Applying Scheduling and Tuning to On-line Parallel Tomography. In *Proceedings of the IEEE/ACM SuperComputing Conference (SC 2001)*, Denver, CO, USA, 2001. IEEE Computer Society Press, Los Alamitos, CA, USA.

[17] W. Sudholt, K. Baldridge, D. Abramson, C. Enticott, and S. Garic. Parameter Scan of an Effective Group Difference Pseudopotential Using Grid Computing. *New Generation Computing*, 22:125–135, 2004.

[18] F. Tamburini. Building distributed language resources by grid computing. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 1217–1220. European Language Resources Association, 2004.

[19] The Annotation Graph Tool Kit. http://agtk.sf.net.

[20] The Natural Language Tool Kit. http://nltk.sf.net.

[21] The UK eScience Programme. http://www.rcuk.ac.uk/escience/.

[22] S. Venugopal, R. Buyya, and L. Winton. A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids. Technical Report GRIDS-TR-2004-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, 2004.

[23] Vijay Pande, et. al.,. Atomistic protein folding simulations on the submillisecond timescale using worldwide distributed computing. *Biopolymers*, 2002.

[24] W.T. Sullivan, III, et. al. A new major SETI project based on Project Serendip data and 100,000 personal computers. In *Proceedings of the Fifth International Conference on Bioastronomy*, Capri, Italy, 1997.

8 Acknowledgements

The research reported in this paper has been supported by the Victorian Partnership for Advanced Computing, the University of Melbourne and the Australian Research Council.