Grid Management Support by means of Collaborative Learning Agents

Wico Mulder
Logica, prof. Keesomlaan 14
PO box 159 1180 AD
Amstelveen
wico.mulder@logica.com

Ceriel Jacobs
VU University Amsterdam,
de Boelelaan 1081 HV
Amsterdam
cjh.jacobs@few.vu.nl

ABSTRACT

The complex and dynamic settings of grid environments lead to challenges on their operational maintenance. The growth of these environments in terms of size and usage requires supporting systems to be of a more sophisticated level. Contemporary tools lack the ability to relate and infer events. Communication across organizational domains and interoperability between existing monitoring tools is subject to improvement. In this paper we present an information system, based on collaborative agents, that supports system administrators in monitoring the grid. While observing log files, the agents learn patterns about job-traffic in their own local domain of the grid and share information to provide global or multi-domain overviews. The agents represent their knowledge in the form of deterministic finite automata (DFA). We discuss our collaborative learning mechanism and show the results of our experiments with data of two grid-sites. Our system generated job-traffic overviews that gave new insights in the performance of the grid environment.

Categories and Subject Descriptors

D.3.3 [**Programming Languages**]: Language Constructs and Features – *Concurrent programming structures.*

General Terms

Algorithms, Management, Reliability, Languages, Theory

1. INTRODUCTION

Over the last few years, grid computing and service orientation have mutually enforced each other. Whereas grid computing provides the technology for sharing large amounts of computational power, service orientation offers flexible and extensible ways of reusing functionalities inside and across organizational boundaries. Together, both technologies gave rise to scalable and configurable service-based computing infra-structures shared by multiple organizations [3].

GMAC'09, June 15, 2009, Barcelona, Spain.

In order to preserve the required service level, the environments in which the services operate need to be maintained carefully. This is a challenge because of the complex and dynamic settings of those environments:

- there are many components and interactions;
- resources may join and leave any time;
- resources are shared by multiple organizations;
- resources are heterogeneous and distributed;
- the components undergo continuous improvements and changing standards.

1.1 Support

While foreseeing strong needs for support of operational maintenance of service-oriented business grid platforms, we are already perceiving maintenance issues in contemporary grid environments. There is a strong need to improve the stability of contemporary production grids. System administrators need to combine information from multiple sources and relate events to find possible causes of a particular problem. Sometimes system administrators have to analyze log files by hand. This is a time-consuming process, certainly when multiple organizational domains are involved.

Our goal is to develop and implement a method for discovering structures in job-traffic in a grid-environment. Using these structures we generate overviews that can be presented to system administrators. In this paper we present a first version of our system. The second goal of this paper is to explain our collaborative learning method. Communication- and privacy constraints prevent grid environments from having all data necessary for analysis situated or accessible from a single place. Therefore, our system consists of a group of individual learners that infer multiple local models and share their models in order to obtain a global model. We show the results of experiments using data of two grid-sites that are part of the EGEE, BiGGrid and LCG environments.

In [8] a learning mechanism for a set of collaborative agents that induce grammars from individual and shared datasets was introduced. We elaborate on this, and discuss two specific approaches for combining models and show how they can be used in the application domain of grid administration.

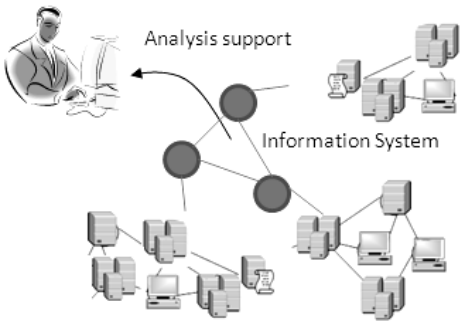

Figure 1, Agents supporting grid administration

The rest of the paper is organized as follows: Section 2 is about related work. In Section 3 we explain our system and our experimental results, Section 4 discusses our approach and algorithm of collaborative DFA learning, Section 5 contains conclusions and Section 6 ends the paper with a discussion and description of future work.

2. RELATED WORK

The field of grid computing contains a variety of monitoring support systems[10]. Whereas some of them focus on supporting infrastructure maintenance and administration (such as Ganglia[1]), others are part of the grid-middleware and support the finding and bookkeeping of resources of interest (such as MDS[2]). Although most of them are designed to be extensible, implementations are often bounded by hierarchy, by organizational boundaries or work primarily with predefined rules.

Communication between components across organizational boundaries and interoperability between the tools themselves is subject to improvement. Furthermore, whereas most tools provide a large variety of overviews, they often lack the functionality of relating events; they are written from the perspective of providing overviews, without the intention to help with problem analysis by means of correlating events. Ganglia e.g., can provide information from many sites at various desired levels of detail, but cannot relate the events in these overviews with each other.

Contemporary grid monitor tools lack the ability to correlate data from multiple sources. This is a practical problem given the growth in usage and complexity of grid environments [7].

Data mining and automated inference of correlations is something that is not yet commonly done in contemporary grid administration tools.

Finding structures in the usage of grid environments means finding patterns, or grammar rules that produce these structures. In this paper we present an effort to model grid job traffic with grammars, in particular Deterministic Finite Automaton (DFA). Application of such algorithms in this dynamic distributed domain is not yet commonly done. Using grammar induction, our system induces and combines traffic patterns from various parts of a grid-infrastructure.

[1] www.ganglia.info
[2] www.globus.org/mds

Recently, a team within the EGEE project started the Grid Observatory initiative[3]. Their aim is to develop a scientific view of the dynamics of grid behavior and usage, making the grid itself more reliable and stable. These efforts are in line with our research.

3. GRID ADMINISTRATION SUPPORT

3.1 CTIS

We developed a system, called CTIS (Collaborative Tracing Information System) that consists of agents[4] that gather job-traffic data from log files and learn DFA models. Together the agents compose overviews that are to be used by grid administrators. CTIS uses a collaborative learning mechanism in which observations result in local models which are shared to obtain a global model (e.g. after each agent has inferred a model from its locally observed data, these models are combined to provide global overviews). Given the dynamic, multi-organizational and decentralized aspects of grid environments, the architecture of CTIS is based on agents[9]. The design of CTIS is based on three principles:

First, to deal with the heterogeneous nature of the grid, the components of CTIS are acting autonomously, fetching the information using local domain privileges.

Second, in order to deal with dynamic aspects, CTIS allows for components (agents) to be added and removed in a flexible and configurable way.

The third principle is that the components of CTIS collaborate in order to retrieve and share information. Communication constraints on the level of bandwidth or privacy imply the need for message control, and abstracting content. The CTIS agents share their individual abstracted models instead of detailed transaction information. Each agent gathers information about job-flows from a system log file. Each agent observes logfiles, uses regular expressions to distill timed event-details (such as e.g. user, organization, time, job_id) and builds DFA models from these event-details.

The task of learning local models is carried out in parallel with the tasks of sending and receiving models. Both tasks are synchronized when merging. In timed intervals the learner is presented with a set of models coming from other learners. Before merging models (temporary considered as hypotheses) into the learners global model, each agent first merges its own local model with its global model.

[3] www.grid-observatory.org
[4] CTIS uses Jade [2], a middleware for the development and run-time execution of peer-to-peer applications which are based on the agents paradigm, see http://jade.tilab.com.

3.2 Environment

We have studied the application of the prototype at the NIKHEF grid-site and the RUG-CIT grid-site. Both sites are part of the of the EGEE[5], BiGGrid[6] and LCG[7] environments.

We have investigated the use of our collaborative learning mechanism in these domains and generated individual and combined models of job traffic information.

Figure 2 shows an example of the situation at two grid sites (domains). Within each domain agents observe different parts of the grid: the compute-element nodes (CE) and a head node of a local batch system (HN). Jobs of a user or group are scheduled by a workflow-management broker (WMF), which sends them to one of the CEs, to be handled by one of the registered organizational domains. The light-gray arrows in the figure reflect the job flows possible in this section of the grid.

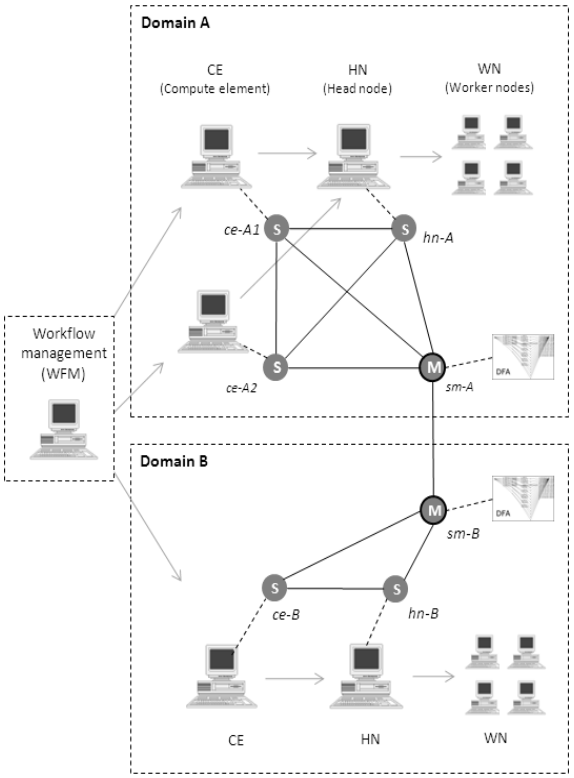

Figure 2. Agents at two grid-sites

The first domain contains two CE nodes and one HN node. The agents at a CE node observe log files from a gate-keeper service, which is part of the grid middleware that accepts jobs for calculation and forwards them to a local batch-system. The agent at the HN node watches accounting log files from a pbs-batch cluster consisting of worker-nodes that handle the actual jobs. The second domain contains one CE - and one HN node.

Since their most primitive function is to observe data from an information source, the agents are called *Sentinels*. Sentinels that operate in the same organizational domain are grouped into a so-called *Squad*. A special agent, the so-called *Squad Manager*, receives the information from the sentinels and creates local DFA models. The squad manager also maintains a global model, which is communicated to other Squad Managers. The moment of communication can be both on timed intervals as well as on the moment of the agents local model update. On request of the system administrator, a squadmanager can provide cross domain traffic overviews based on its global model.

3.3 Resubmitted jobs

Of interest to system administrators is an overview of resubmitted jobs, i.e. jobs which are not handled properly and are scheduled more than once. Figure 3 (next page) shows a graph showing (generic) paths of jobs that didn't pass a CE node and were resubmitted. The links show the transitions from one node (logical machine) in the grid to the next one. E.g. a job was on a machine of CERN, went through CE named Gazon, blocked but continued via the HN, cluster-group 'lui2' and finally ended on worker-node '007'. The nodes in the graph represent the state of the grid inside the nikhef-domain.

The picture shows that the jobs were resubmitted at CE level and continued via HN and were handled by a worker-node. By combining DFA models from two domains, our system generated supporting overviews of resubmitted jobs.

These overviews support the system administrators in detecting the possible causes and show whether alternative paths (through other organisational domains) were followed instead.

3.4 Cross-domain traffic patterns

We used the global models of the agents to get insight in the job traffic behavior across two domains. Figure 4 (next page) shows an example of an obtained model that shows the generic actual job-flow structure over both the NIKHEF grid site (CE Gazon, Trekker) and the RUG-CIT grid site.

These type of overviews provide useful insights in the behavior of a grid as whole (or at least larger multi-domain parts of it). Note that the DFA folding (MDL) causes some confusing arrows back to the starting node. For human interpretations of this graph, this can be an annoying artifact, which must be improved.

---

[5] Enabling Grids for E-Science in Europe (EGEE), a European project. See also http://www.eu-egee.org.

[6] BiGGrid is the Dutch e-Science Grid. See also http://www.biggrid.nl.

[7] The LHC Computing Grid (LCG) is the grid environment that supports the particle physics experiments using the Large Hadron Collider (LHC) at CERN. See also http://gridcafe.web.cern.ch.

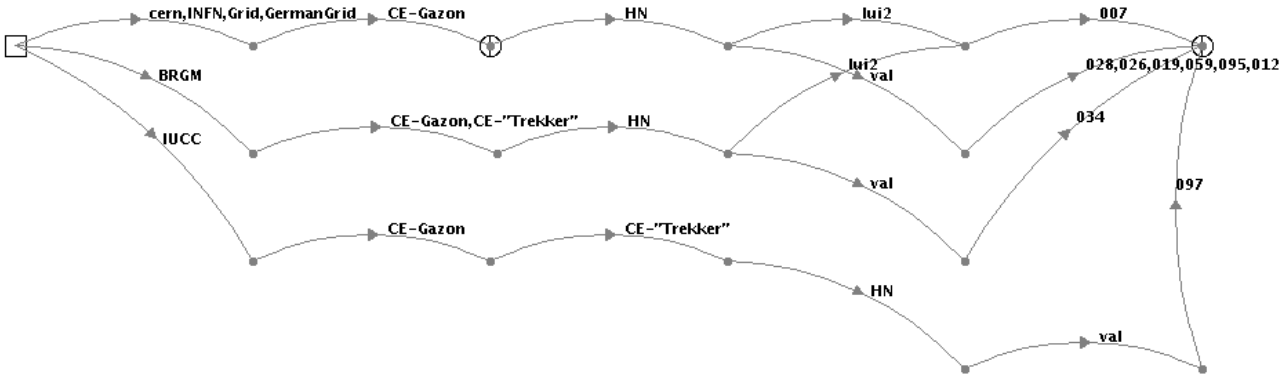

Figure 3. Resubmitted job traffic DFA graph.

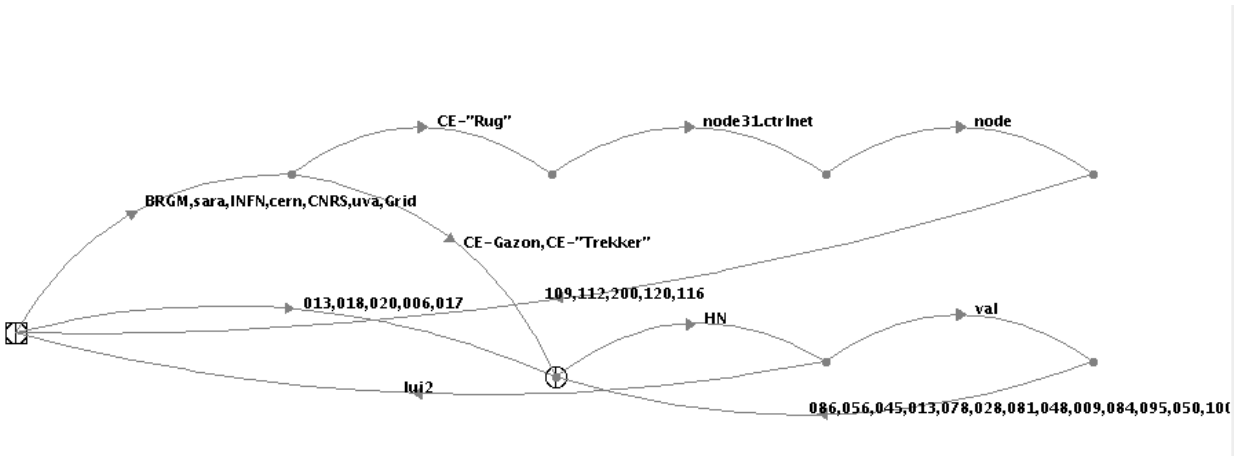

Figure 4. Traffic overview across two domains.

4. COLLABORATIVE LEARNING

The task of finding patterns in grid-traffic data over multiple domains is regarded as a collaborative learning task carried out by a set of learners.

4.1 Grammar induction

Our approach is based on grammar induction. A learning algorithm is used to obtain a grammar that explains the structure of a given set of data. The aim is to learn from sample data (usually a list of sentences) an unknown grammar which explains this data. The model is also used to verify whether unknown samples follow the rules of the grammar.

A Deterministic Finite Automaton (DFA) is a type of model that is commonly used to classify a structure (language) and represent a grammar in the form of a graph[5]. A DFA can be seen as a model which captures the underlying rules of a system, from observations of its behavior or appearance. These observations are often represented by sentences that are labeled "accepted" or "rejected". Every sentence in the dataset is represented as a path in the graph. Since creating some DFA that is consistent with training data is trivial, it is usual to add the constraint that the DFA should generalize to unseen test data.

Since our sentences are observations from sentinels, in the following, we only learn from sentences that are to be accepted.

4.2 Individual DFA Learning

The individual DFA learning mechanism starts with a learning sample, which is a set of sentences that are supposed to be accepted by the target DFA. First, it builds the prefix tree acceptor (PTA) for the sentences. This is the (tree-shaped) DFA that accepts exactly the sentences in the learning sample, and nothing else. Then, it uses heuristics to generalize, by merging states and making the resulting automaton deterministic again by applying further merges. The algorithm used to determine the merge candidates is blue-fringe [6]. This algorithm maintains a core of "red" states, which are states of the final DFA, and a list of "blue" states, which are the states that are considered for either to be promoted to "red" or to be merged with a red state. This set of "blue" states consists of all children of "red" states that are not

themselves "red" states. The "blue" states are the heads of sub-trees of the PTA. The algorithm uses Minimum Description Length (MDL) [4] to decide on the next step (either to promote a "blue" state to "red" or to merge a "blue" state with one of the "red" states). To decide on the next step, all possibilities are tried, the MDL of each of the results is computed, and the best one is chosen, after which this whole process is repeated, until all states are "red" states.

MDL consists of the sum of two parts: the number of bits needed to encode the model (model-code), and the number of bits needed to encode the sample, given the model (the data-to-model code). The lower the sum, the better the model is assumed to be. We will not discuss the model encoding here. For details, the reader is referred to Adriaans et. al[1]. The data-to-model code is computed as follows: suppose *l* is the maximum length of a sentence in the sample. It is possible to compute the total number of sentences *N* with length less than or equal to *l* that are accepted by the model. Also, the number of sentences S in the sample is known. Obviously, this set of sentences is a subset of the set of all accepted sentences with length less than or equal to *l*.

Now, there are $p = \binom{N}{S}$ ways to select S sentences from N sentences. So, an integer with possible values between *0* and *p-1* uniquely identifies the sample, thus for the data-to-model code we need *2log(p)* bits. Note that promoting a "blue" state to "red" does not alter the MDL. Therefore, states are only promoted to "red" if all attempted merges result in a worse MDL score.

4.3 Local and Global models

We consider a large distributed dataset and a set of learners that have the collaborative task to model this dataset. We call this dataset the global dataset. Each learner can only observe a part of the global dataset. This part, called the local dataset, contains data that belongs to the local environment of the learner. The global dataset represents e.g. job-traffic data of the grid-environment. A local dataset represents e.g. job-traffic within a particular organizational domain.

Figure 5 shows two learners that observe (parts) of traffic data in a grid environment. Each learner observes a local dataset and shares its global models. Each learner builds a model, called its local model, that reflects the grammatical structures in its local dataset.

In order to fulfill the collaborative learning task, the learners have to share and merge their models. Our application environment however does not allow all details from one domain to be communicated to another domain. Our strategy is therefore to let each learner maintain a second model. This model, called the learner's global model, contains only information that is allowed to be shared across domains.

The global model of a learner is communicated to other learners. When a learner receives a model from another learner, it is treated as an hypothesis and merged with the global model of the receiving learner.

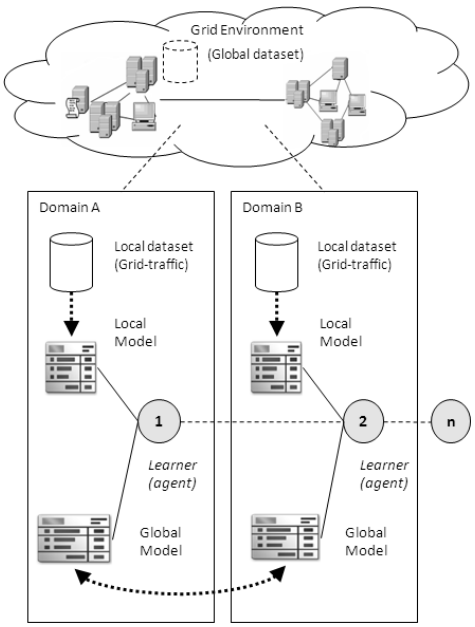

Figure 5, Two agents analyzing grid-traffic data. Each agent observes data from its own local environment.

By means of sharing and merging their own global DFA models, the agents are able to provide a total model of job-flows on the grid.

An interesting aspect of this mechanism is that it allows each learner to have its own global model, and these models are allowed to be slightly different from each other (like in Plato's theory of Forms). Although it is common to share global information between agents using a central blackboard, our mechanism allows the learners to have their own global model which, in fact, can be regarded as an implicit, redundant, distributed blackboard. On one hand this approach involves extra costs, but on the other hand it allows for autonomous interpretations about and use of the model.

4.4 Collaborative DFA Learning

The process of collaborative learning is characterized by how and when models are merged. Furthermore it must be specified how to continue learning with a merged model.

We use two different learning methods: complementary DFA learning, in which models from different environments are combined (stitched) into a single one, and congruent DFA learning where models of similar environments are merged. In the process of complementary learning, the samples of the different models have different structures. After learning each individual model, they are combined by means of overlapping head-tail nodes. In the second case, the models are based on similar structures, and the merge of these models allows for an enriched combined model on the level of its details.

In complementary DFA learning, two models from different environments are combined as follows: if the first DFA has an edge to an end-state on symbol s, and the second DFA has an outgoing edge from its start-state on that same symbol s, the two

edges are replaced by a single edge from the source of the first edge to the destination of the second edge. If the destination of the first edge has outgoing edges, these are added to the destination of the second edge. In general, this may result in a non-deterministic finite-state automaton, which can be made deterministic by applying a subset algorithm [5].

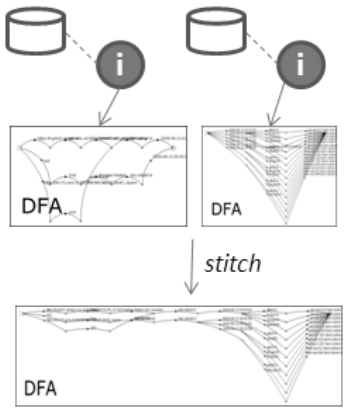

Figure 6. Complementary Learning

In congruent DFA learning, two DFAs from similar environments are merged by combining the start-states of both DFAs. This may result in a non-deterministic finite-state automaton, which again can be made deterministic by applying a subset algorithm. In general, the application of a subset algorithm may result in a considerable increase of the number of states of the resulting DFA. It therefore makes sense to attempt to apply a learning step on the resulting DFA.

However, the blue-fringe algorithm is no longer applicable. Instead, a general learner can be applied that just tries to merge all state-pairs of the DFA, repeatedly, until no more improvements are found, according to the MDL scores. So, we again need an MDL score for the DFA resulting from the merge. Unfortunately, this score is not readily available, because we must assume that the samples from which the original DFAs were learned are not available anymore.

So, we need an estimate of the MDL score of the resulting DFA.

At first sight, one might think that the original sample sizes could just be added to obtain a new sample size for the resulting DFA, but this is not the case. In congruent DFA learning, there might be overlap in the samples, and in complementary DFA learning, adding the sample sizes makes even less sense. In fact, we will have to estimate a sample size for the resulting DFA.

In Section 4.2 we discussed how the MDL score is computed. In particular, the MDL score does not depend on the particular sample, but rather on the sample size. In fact, we can assume that the sample size (the S) and the maximum sample length (the l) are available for the original DFAs, and from these numbers we can compute a "sample density", which is S divided by the total number of accepted sentences with length less than or equal to l (the N).

Now, let S_1 and S_2 be the sample sizes of the original DFAs, and l_1 and l_2 be the maximum sample lengths, and let D_1 and D_2 be the sample densities. An estimate for the maximum sample length l of the resulting DFA is obtained as follows: for congruent learning, we use the maximum of l_1 and l_2, for complementary learning we use $l = l_1 + l_2 - 1$. For the sample density D of the resulting DFA we use the average of D_1 and D_2.

To compute the estimated sample size S of the resulting DFA we compute the total number of accepted sentences with length less than or equal to l, and multiply this number by D.

This computation allows for the computation of an MDL score, and thus allows for the application of another learning iteration on the resulting DFA.

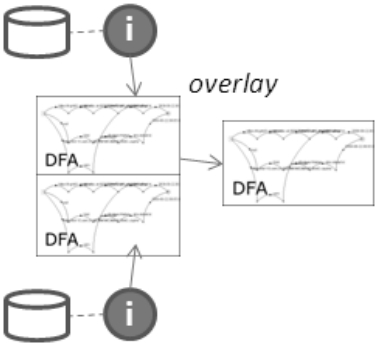

Figure 7. Congruent Learning

5. CONCLUSION

In this paper we introduced CTIS, a distributed information system that supports grid administrators with cross-domain job-flow patterns.

We worked out a collaborative learning mechanism and discussed how it attempts to meet the communication constraints on the exchange of individual data. We applied our mechanism on grid job traffic data, and showed that complementary and congruent merges provide useful overviews.

Whereas complementary learning was used to obtain global models representing job traffic on successive nodes, congruent learning was used to obtain a global model of user traffic that is spread over similar nodes in parallel.

We focused on the application of our system in a real-life grid environment. Using data from this environment, we studied the application of our collaborative DFA learning mechanism.

The CTIS system complements existing support provided by other monitoring software and focuses on correlation of information. The system is designed to act in a dynamic, distributed environment, and to share information across multiple organizational domains.

Using the mechanism of sharing DFA models across multiple organizational domains, our system generated useful overviews of resubmitted jobs and cross-domain patterns in job-traffic data.

6. DISCUSSION

6.1 Consistency checks

In experimental settings we were able to neglect some of the constraints allowing us to model the global dataset as-a-whole by means of a single learner. Such a single-learner model can be obtained in ideal situations, when a single agent is allowed to access the global dataset.

The question is if the global model at each agent, obtained by merging hypotheses, converges to the single-learner model, or whether it just approximates it.

To check whether the collaborative learning mechanism provides the same models as single learners could have done, we compared the content of combined DFAs with a separately learned DFA obtained by a single process that observes all individual data sources at once. For each DFA we calculated a score (the MDL score) that represents the number of bits necessary to describe the structure of the model together with the number of bits necessary to describe the sample data given this model. While increasing the period in which the job traffic is observed, we compared the scores of the combined DFAs with the single learned DFAs. The scores of the combined DFA and the single ones were in these cases the same; sometimes differing slightly due to 'open stitch ends', meaning that some of the endpoints of the DFA from the CE agent do not match with the starting points of the DFA of the HN agent or vice versa. This can be explained by the fact that some jobs at the CE are not handled yet by the HN, or because of resubmitted jobs.

6.2 Future Work

Although CTIS is not yet comparable in terms of size and functionality with other monitoring systems, the approach of collaborative learning agents distinguishes it from other systems. Collaboration combined with learning provides new insights in how information from different domains in a grid can be correlated.

Whereas we currently focus on monitoring, we foresee possible roles for CTIS in grid-management in general. The current prototype is designed for job analysis and support in grid environments, but the concepts of CTIS can also be applied to other (networked) environments. We intend to use CTIS for monitoring services on application- or process levels, in which traffic can be related to service level agreements (SLA). In its current status, we use the prototype to perform feasibility studies on its use for grid administration. Within this application domain we want to further investigate the usage of our collaborative learning approach. Examples are cases of analyzing 'black hole worker nodes' or anomaly detection based on suspected patterns (intrusion, malicious usage).

Foreseeing the growth of commercially oriented service platforms, we plan to further study the concepts and applications of CTIS in these areas.

7. Acknowledgments

We thank the Dutch institute for high energy physics, NIKHEF, and the Donald Smits Center for Information Technology (CIT) at the university of Groningen for their support and access to the data of a representative grid environment.

This work was carried out in the context of Virtual Laboratory for e-Science project (www.vl-e.nl). This project is supported by a BSIK grant from the Dutch Ministry of Education, Culture and Science (OC&W) and is part of the ICT innovation program of the Ministry of Economic Affairs (EZ).

We like to thank Maarten van Someren for his constructive comments. We also like to thank Eduard Drenth and Arjan Stoter from the Logica CNS-Java team for their development support and general comments.

.

8. References

[1] Adriaans, P., Jacobs, C.,"Using MDL for grammar induction", in Grammatical Inference: Algorithms and Applications, 8th International Colloquium, ICGI 2006, pp. 293-306.

[2] Bellifemine, F., Ciare, G., Greenwood, D., "Developing multi-agent systems with JADE", John Wiley & Sons, 2007, ISBN 0470057475

[3] Boss, G., Malladi, P., Quan, D., Legregni L., Hall H., "Cloud computing", IBM Whitepaper Oct 2007, http://download.boulder.ibm.com/ibmdl/pub/software/dw/wes/hipods/Cloud_computing_wp_final_8Oct.pdf

[4] Grünwald, P., "A tutorial introduction to the minimum description length principle". In: Advances in Minimum Description Length: Theory and Applications (edited by P. Grünwald, I.J. Myung, M. Pitt), MIT Press, 2005 (80 pages).

[5] Hopcroft, J.E., Ullman, J.D., "Introduction to Automata Theory, Languages, and Computation", Addison-Wesley, Reading, Massachusetts, 1979.

[6] Lang, K.J., Pearlmutter, B.A., Price, R.A., "Results of the Abbadingo One DFA learning competition and a new evidence-driven state merging algorithm", in Grammatical Inference; 4th International Colloquium, ICGI-98, Springer LNCS/LNAI #1433, pp. 1-12, Ames, Iowa, USA, July 1998.

[7] Michaud, L., "Intelligent agents for network management", Computer Science Department Institute of Computer Communications and Applications, École Polytechnique Fédérale de Lausanne, June 1998.

[8] Mulder, W. Meijer G. R., Adriaans P., "Collaborative learning agents supporting service network management", in Service-Oriented Computing: Agents, Semantics, and Engineering, AAMAS 2008 International Workshop, SOCASE, in Springer LNCS 5006 pp83-92,ISBN 978-3-540-79967-2.

[9] Stone, P., Veloso, M., "Multiagent systems: A survey from a machine learning perspective", Autonomous Robots, 8(3):345–383, July 2000.

[10] Zanikolas, S., Sakellariou, R., "A taxonomy of Grid Monitoring systems", Science Direct, Elsevier, 2005.