A Catallactic Market for Data Mining Services

L. Joita[a], Omer F. Rana[a], Felix Freitag[b], Isaac Chao[b], Pablo Chacin[b], Leandro Navarro[b] and Oscar Ardaiz[c*]

[a]School of Computer Science, Cardiff University, UK

[b]Computer Architecture Department, Technical University of Catalonia Jordi Girona 1-3, Campus Nord D6 Barcelona, Spain

[c]Department of Mathematics and Informatics, Public University of Navarra, Campus de Arrosadia, Pamplona, Spain

We describe a Grid market for exchanging data mining services based on the Catallactic market mechanism proposed by von Hayek. This market mechanism allows selection between multiple instances of services based on operations required in a data mining task (such as data migration, data pre-processing and subsequently data analysis). Catallaxy is a decentralized approach, based on a "free market" mechanism, and particularly useful when the number of market participants is large, or conditions within the market change often. It is therefore particularly suitable in Grid and Peer-2-Peer systems. The approach assumes that the service provider and user are not co-located, and require multiple message exchanges to carry out a data mining task. A market of J48-based decision tree algorithm instances, each implemented as a Web Service, is used to demonstrate our approach. We have validated the feasibility of building catallactic data mining grid applications, and implemented a proof-of-concept application (Cat-COVITE) mapped to a Catallactic Grid Middleware.

1. Introduction

In a service rich environment where multiple instances of a given service are available, identifying how a selection can be made has been a topic of significant research – especially in Grid computing [15]. Generally, such a discovery is supported by the availability of specialist registry services, which allow metadata about a particular service to be recorded. As an alternative, economic models allow the selection of Web/Grid Services based on a market mechanism (such as auctions), and become useful as a greater number of such services become available [3]. Most existing approaches generally rely on a centralized broker that coordinates resource access in a market [18]. We propose an approach based on the Catallaxy mechanism of von Hayek [6], which is decentralized and therefore does not require a centralized broker.

The Catallaxy approach is a coordination mechanism for systems consisting of autonomous decentralized service users and providers (agents), and based on negotiation and price signalling between such agents [4]. Catallaxy makes use of a "free market" approach, allowing prices for services to be altered based on demand. The use of Catallaxy therefore leads to the development of self-organizing individuals (agents) that are highly dynamic, thereby leading to systems which behave in a Peer-2-Peer fashion. Such an approach is particularly suited to "Open Systems", where detailed knowledge about particular agents may not be known *apriori*.

We propose the use of such a Catallactic market for exchanging data mining services, previously described in [16]. These services constitute algorithms that are part of the WEKA toolkit [14], and which we have converted to Web Services. Approximately 75 different algorithms (primarily classifiers, clustering algorithms and association rules) are provided. Additional capability to

*This work has been supported in part by the European Union under Contract CATNETS EU IST-FP6-003769, and the Spanish Government under Contract TIC2002-04258-C03-01

support attribute search and selection within a numeric data set is also provided, with 20 different approaches to achieve this (such as the use of a genetic search operator). We describe how a market for data mining service may be established, by the use of a Catallactic middleware that can interact with computational resources hosting these services. Existing Grid users can therefore utilize our approach without significant modifications to their existing applications, through the use of a market access point – described further in section 4. As a key contribution in this work, we map the market concept to operations supported in data mining, and demonstrate the feasibility of building a Catallactic data mining market. Subsequently, a proof-of-concept application (Cat-COVITE), mapped to a Catallactic Grid Middleware, is used to demonstrate our ideas.

2. Data Mining Grid Markets

We consider a Grid environment to consist of a "resource" and a "service" market. In a resource market, providers sell processor capacity, storage, or bandwidth. Such a market involves physical resources which can host one/more services. Conversely, in a service market providers sell application services. A collection of such services composed together may constitute an application. A given service may be hosted on different resources, and each service instance would have a different price (for instance, a service executed on a single processor machine may cost less than one hosted on a parallel machine). If a large number of service sellers and buyers exist, market mechanisms may be used to chose between them. Moreover, service providers can buy resources at the Grid resource market to provide services in the Grid service market. Both markets are therefore dependant on each other, but can operate autonomously applying Catallactic mechanisms as has been shown by simulations in [2].

The basic problem addressed by the data mining process is one of mapping low-level data (which are typically too voluminous to understand) into other forms that might be more compact (for example, a short report), more abstract (for example, a descriptive approximation or model of the process that generated the data), or more useful (for example, a predictive model for estimating the value of future cases). At the core of the process is the application of specific data-mining methods for pattern discovery and extraction. This process is often structured into a discovery pipeline/workflow, involving access, integration and analysis of data from disparate sources, and to use data patterns and models generated through intermediate stages. A particular use of Grid computing in this context would be to combine services that implement specific phases in the discovery pipeline, which includes:

- Selection of a data set. The data set may be in a variety of different formats (such as Comma Separated Values, Attribute Relation File Format, etc), and often converters may be necessary. Let $T_c(ds)$ represent the time to achieve this conversion and any preprocessing required on the data – such as support for selection of attributes that are part of the data set. This time is a function of the size of the data set (represented by variable ds). This task may be undertaken at the client or at the service provider.

- Selection of a data mining algorithm. This selection depends on the nature of the data and knowledge to be extracted from the data set. The selection process could be automated through the use of pre-defined rules, or based on the past experience of a user. This stage may be skipped if the user already knows which algorithm is required. Making this choice is often a difficult decision to make, and generally little support is provided in existing tools – a user is often presented with available algorithms, and has to make a choice manually. Let $T_{as}(n)$ represent the time to find a suitable algorithm from n possible algorithms.

- The third stage involves the selection of the resources on which the data mining algorithm needs to be executed. The choice may be automated by the underlying resource management system, if multiple instances of the same algorithm can be found, or the

user needs to manually request access to a particular algorithm on a specific resource. Let $T_{rs}(m)$ represent the time to find a suitable resource from m possible resources or resource bundles.

- The data mining algorithm is now executed - by checking security constraints on the remote resource, and often the data set may need to be migrated to the remote resource. Let $T^a_{md}(ds)$ represent the time for data migration, T_c represent the time to configure the service instance on a particular resource (such as checking of security credentials), and T_e represent the time to execute the algorithm.
- The generated model from the data mining algorithm is now presented textually or graphically to the user. The model may now be verified through the use of a test set. Let $T_{pp}(ds)$ represent the time to achieve this post processing on the data, and $T^b_{md}(ds)$ the time to migrate the data back to the client.

Hence, the total time required to undertake a particular data mining task is:

$$\tau = T_c(ds) + T_{as}(n) + T_{rs}(m) + T^a_{md}(ds) \quad (1)$$
$$+T_c + T_e + T_{pp}(ds) + T^b_{md}(ds)$$

assuming $T_c(ds) \approx T_{pp}(ds)$, $T^a_{md}(ds) \approx T^b_{md}(ds) = T_{md}(ds)$, and $T_c \ll T_e$, then $(T_c + T_e) \approx T_e$ – hence we can simplify τ to:

$$\tau = 2(T_c(ds)+T_{md}(ds))+T_{as}(n)+T_{rs}(m)+T_e \quad (2)$$

The use of the Catallactic market is primarily intended to associate a cost with $T_c(ds), T_{md}(ds)$ and T_e. A service provider that has a high bandwidth is able to provide a fast pre- and post- processing service. Similarly, having a low execution time for the analysis algorithm would have the highest cost in the market. The times $T_{as}(n)$ and $T_{rs}(m)$ are associated with the service and resource markets respectively. The Catallactic middleware is intended to mimic the behaviour of a Catallactic market, and needs to be efficient enough to be able to reduce these times compared to other market mechanisms.

We have assumed that pre- and post-processing of data is undertaken by the client or service provider, although it is also possible for such processing to be undertaken by a different service. Assuming that there are k services in a pipeline, with each stage being undertaken by different service and resource providers (representing data pre-processing and transformation, analysis, post-processing and visualization) we can represent, as an upper bound, a time of $(k \times \tau)$ to represent the total time for the data mining task. It would now be necessary to engage the Catallactic middleware k times to find suitable service instances in the market.

3. Cat-COVITE Prototype

To demonstrate our ideas, an existing application called the Catallaxy COllaborative VIrtual TEams (Cat-COVITE) [9] has been extended. Cat-COVITE is based on a Service-Oriented architecture, and consists of three main elements: (i) one or more user services; (ii) a "Master Grid Service" (MGS) – responsible for interacting with a Catallactic middleware to find an end point reference for a service instance, and (iii) one or more service instances that are being hosted on a particular resource. Cat-COVITE currently supports searching through distributed product catalogues (each being a database wrapped as a Web Service), achieved by launching multiple concurrent searches. This database search has been extended with data analysis services which may operate on data sent by a client service, or data already available where the analysis service is hosted.

Previous work has involved translating data mining algorithms supported in the WEKA toolkit into Web Services – a number of classification and clustering algorithms have been converted [16]. The Catallaxy COllaborative VIrtual TEams (Cat-COVITE) prototype makes use of classifier services that implement a J48 decision tree classifier, based on the C4.5 algorithm [5].

The J48 service has two options: classify, and classify_graph. The classify option is used to apply the J48 algorithm to a data set specified by the user. The data set must be in the ARFF format, which essentially involves a description of a list of data instances sharing a set of attributes. The result of invoking the classify operation is a textual output specifying the classification decision tree. The classify_graph option is similar to the classify option, but the result is a graphical representation of the decision tree created by the J48 service. We therefore have a market of J48 services, each being hosted on different resources.

3.1. Use of WS-Agreement

Web Service Agreement (WS-Agreement) by the Grid Resource Allocation and Agreement Protocol Working Group (GRAAP WG) provides a protocol for specifying an agreement between a resource/service provider and a consumer [12]. It is generally aimed to be a one-shot interaction, and does not support negotiation. However, it can form a useful basis for describing an agreement once negotiation has been conducted using other approaches (negotiation support is also currently being investigated as an extension).

WS-Agreement is used in Cat-COVITE, and forms the basis for choosing between multiple service and resource providers. The service provider acts as the agreement provider, while the service consumer as the agreement initiator. When using WS-Agreement in our prototype, several parts need to be specified [12]: agreement name, the agreement context – parties to the agreement, reference to the service(s) provided in support of the agreement, and the lifetime of the agreement. Agreement terms, which describe the agreement itself, can contain: the service description terms, which provide information needed to instantiate or otherwise identify a service to which this agreement pertains. Finally, guarantee terms which specify the service levels that the parties are agreeing to. An example of an Agreement and Offer template used by Cat-COVITE is provided in the appendix.

4. Catallactic Grid Middleware

The implementation of Catallaxy in real world Grids requires the design of Catallactic middleware which offers a set of generic negotiation mechanisms, allowing specialized strategies and policies to be dynamically added as plugins. It is intended that the middleware offers a set of high level abstractions and mechanisms to locate and manage resources, locate other trading agents, engage agents in negotiations, and adapt to changing conditions. A detailed description of middleware design process, including requirements analysis, layered architecture and implementation details can be found in [17].

At the middleware layer, a set of agents provide the capabilities to negotiate for services and the resources needed to execute them. We differentiate between a "Complex" service and a "Basic" service – to determine how the resulting market should be structured. A Complex service is essentially composed of other services, and may not be available on a computational resource. A user application is an example of a Complex service – which may involve services directly owned by the user, and those that may be acquired from elsewhere. A Basic service may be hosted on a particular resource, and is made available on the Catallactic market. A Basic service is therefore the key element that is traded on the Catallactic market – although a Basic service can have various instances when hosted on different resources. A Complex Service agent acting on behalf of the application initiates the negotiation. Basic Service and Resource agents manage the negotiation for services and resources, respectively

4.1. Data Mining Cat-COVITE Services and Catallactic Grid Markets

Figure 1 shows the prototype components and related Catallactic agents as buyers and seller in the Grid service market and Grid resource market. The prototype is composed of three main components, the `MasterGridService (MGS)` – as a type of Complex Service, the Data Mining Services – as types of Basic Services, and job execution resources - as computational resources. The `MGS` Complex Service is the buyer entity in the

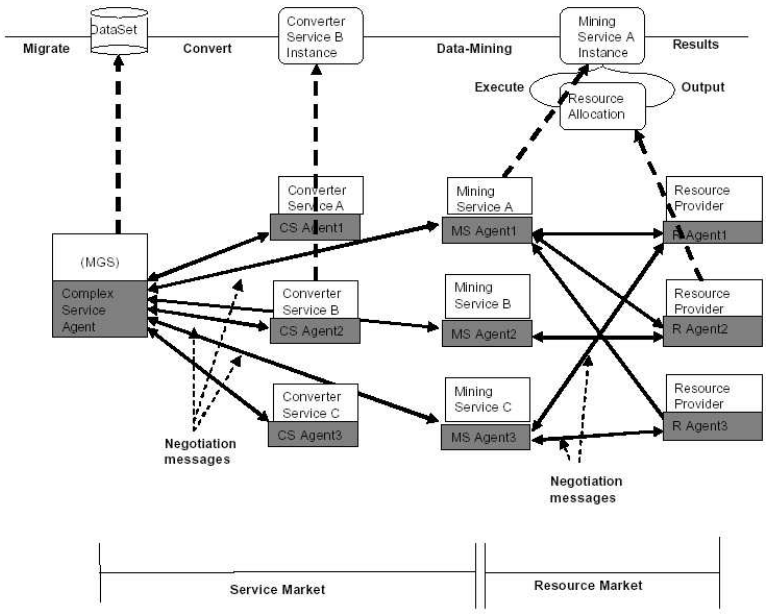


Figure 1. *Logical Components in the Cat-COVITE market for Data Mining services*

service market, and the Basic Service is the seller entity in the service market. A `MGS` forms the key element in the Cat-COVITE prototype, and undertakes the following activities:

- Translates a request to a Basic Service – in this instance such a service is a data mining service.
- Starts multiple sessions with a number of agents that are responsible for finding Basic Services. Each session involves a negotiation to find a suitable Basic Service.
- Passes an end point reference of the Basic Service to the client for invocation.

The Basic Service involves data mining job execution and consists of a data mining Job Execution Environment, which offers the deployment of multiple "slaves" able to execute the data mining task. Within the Cat-COVITE prototype, the data mining Basic Service needs to be able to provide a response time as an important characteristic – this is equivalent to a sum of parameters $T_c(ds) + T_{md}(ds) + T_e$ from equation 1 in Section 2. With this goal the data mining Basic Service buys resources in the resource market. Resource seller entities are able to provide a set of resources via the Local Resource Manager (LRM). The Resource Agents act on behalf of these Local Resource Managers (LRMs), and provide an interface to the physical resources they manage. The data mining Basic Service is the buyer entity in the resource market, and the Resource Local Managers are the seller entity on the resource market. The main functionalities of Basic Service agent at the resource market are: (i) co-allocation of resources (resource bundles) by parallel negotiation with different resource providers (local resource manager entities); (ii) informing the Complex Service about the outcome of resource negotiation.

The Cat-COVITE data mining scenario involves an `MGS` which needs to run a data mining job. The `MGS` sends an `AgreementOffer (AO)`, based on the `AgreementTemplate (AT)` downloaded from the `CatallacticAccessPoint`

(CAP), to the CAP to find a data mining service. The CAP is a Web Service located on a machine providing a catallactic market access point, acts as a WS-Agreement provider for the application and as a factory for Complex Service Agents which will start the negotiation process. The CAP therefore provides an entry point into the market, and can allow existing Grid applications to make requests directly to it. The Complex Service Agent, acting on behalf of the MGS (as a complex service) chosen by the CAP, negotiates with the Basic Service Agents (in the Cat-COVITE markets environment) for data mining services. The AT specifies the service properties that are necessary to create an instance of a service using a factory service. The AT and AO are provided in the Appendix. The DecisionMaker, which is part of the CAP, takes the decision of accepting or rejecting the agreement offer sent by the MGS. There are multiple factors involved in this decision, such as: the parameters in the agreement offer that are part of the agreement template, and the possibility of finding the available Basic Service(s) within the Catallactic market at the budget specified by the MGS as service requestor.

4.2. Implementation

Figure 2 shows the placement of logical components along the three layers: the application layer, the Catallactic middleware layer and the base platform layer. At the application layer, an interface must be provided to issue the requests for services to the middleware, and use the references to service instances provided in response. At the middleware layer, a set of agents provide the capability to negotiate for services and resources. The Complex Service agent acts on behalf of the application and initiates the negotiation. Basic Service and Resource agents manage the negotiation for services and resources respectively. A Service Factory is provided to instantiate the service on the hosting platform selected during the negotiation process. Finally, at the Base Platform layer, a Resource is created to manage the allocation of resources to the service. This resource represents the "state" of the service from the perspective of the middleware (however this does not mean the service is stateful from the perspective of the application). The flow of information among the logical components can be summarized as follows: a Client issues a request to the application (1), which builds a data mining job and requests the execution of this job to the MGS (2). The MGS contacts a CAP asking for a WS-Agreement template for such a service. The MGS fills in the template and sends back an AO (3). The Complex Service Agent initiates the Catallactic mechanism to find appropriate Basic Services and Resources. The Complex Service Agent uses the discovery mechanisms implemented in the middleware to locate Basic Service Agents providing the J48 Service. When a number of Basic Service Agents are discovered, it starts negotiations with one of them (4). In turn such Basic Service Agent must discover and negotiate with a Resource Agent for resources(5). Negotiations are implemented by the Economic Framework Layer, where different protocols can be used depending on the agent's strategy. When an agreement with a Basic Service Agent is reached, the Resource Agent instantiate a Resource to keep track of the allocated resources and returns to the Basic Service Agent a handle for this resource (6). Consequently Basic Service Agents use the service Factory to instantiate the J48 service on the selected GT4 container (7). Basic Service Agent returns to the Complex Service Agent the End Point Reference (EPR) to this J48 service instance (8), forwarded to the MSG (9), which uses it to invoke the service (10). The CAP therefore provides an interface between an application wishing to discover suitable services, and the underlying resources that host those services.

4.3. Physical Deployment on GT4 containers

The logical architecture from the previous section can be implemented in different ways depending on the base platform used. We describe a specific implementation using Globus Toolkit 4 (GT4). We assume the services are previously deployed on a set of GT4 containers. The only resource properties considered in the negotiation are the access rights to execute the service on a specific container. Finally, the service can be instantiated on a container using a generic fac-

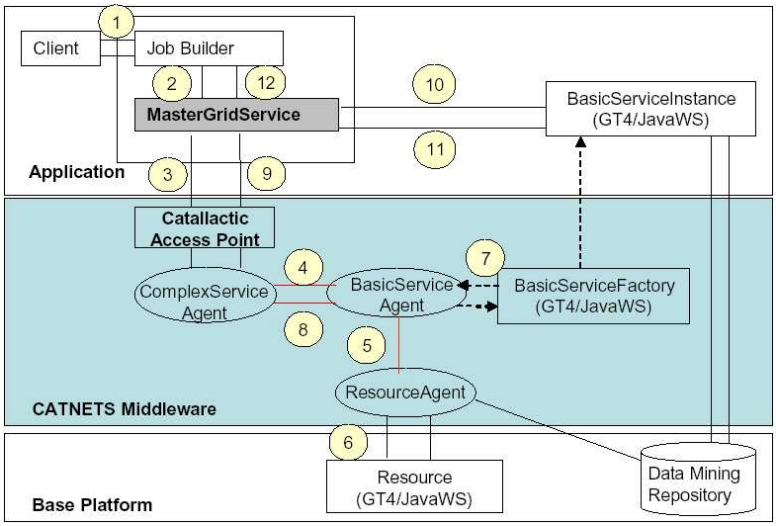


Figure 2. *Interaction between application, Catallactic middleware and computational resources*

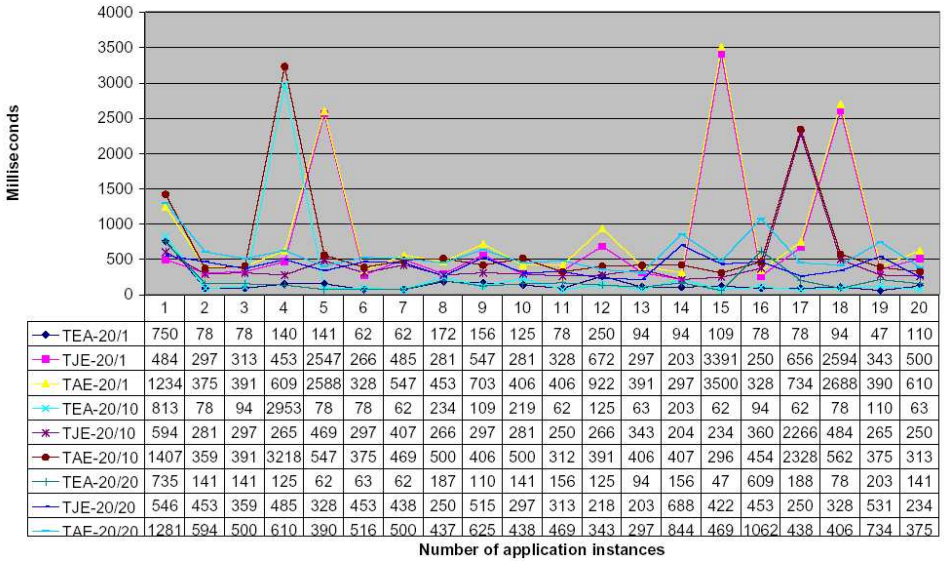
tory. The user application resides in a host (or series of hosts), which also provide the `MGC` and the Complex Service Agent, which represents the application in the negotiation process. On each Grid Container (GT4) where the Data Mining Job Execution Service is deployed, resides the corresponding Basic Service Agent, which negotiates with the Complex Service Agent for access to the Data Mining Job Execution Service. In this container also resides the Resource Agent, which negotiates with the Basic Service Agent for the rights to execute the Data Mining Job Execution Service in this container. Finally, a Resource is created as result of the negotiation process, which represents the rights to execute the service in this container.

5. Evaluation

Our experimental testbed consists of a WinXP machine at Cardiff and Linux RedHat 7.3 at UPC (Barcelona) and Pamplona. Each site also provide a GT4.0 installation and an Apache Axis 1.2/Tomcat 5.0.28 installation at UPC (Barcelona). Each of these machines also host a GT4.0 container. The application prototype settings are as follow: there are 10, 20 and 30 instances of a Cat-COVITE application, each of which requires access to a data mining service (a J48 Web Service) and one `CAP`. The `Master Grid Service` is hosted in Cardiff, and the `CAP` in Barcelona. J48 service instances are made available in Cardiff, Barcelona and Pamplona – via GT4.0 containers. We assume in this case that the user knows that they want to make use of the J48 service, but multiple instances of this service exist. There is a simulation agent environment acting on behalf of services and resources.

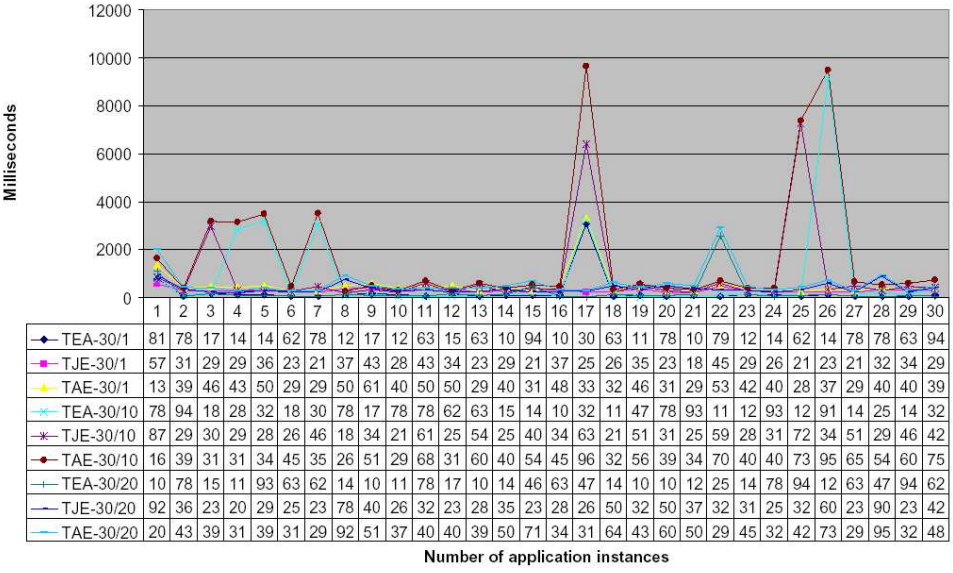
Figures 3 and 4 illustrate the time distribution for exchanging the agreements between the instance applications (via `MGS`) and `CAP` (labelled TEA in the figures), the time for job execution (labelled as TJE in the figures), as well as the time for executing the user application (labelled as TAE in the figures) – all of these are measured at the application level. It may also be useful to note that: TJE is represented by the time between events (10) and (11), TEA by time between events (3) and (9), and TAE by time between events (2) and (12) in figure 2.

In figure 3, the time to discover a service (equivalent to $T_{as}(n) + T_{rs}(n)$ – in Section 2) is represented by TEA-20/*. This experiment in-



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
TEA-20/1	750	78	78	140	141	62	62	172	156	125	78	250	94	94	109	78	78	94	47	110
TJE-20/1	484	297	313	453	2547	266	485	281	547	281	328	672	297	203	3391	250	656	2594	343	500
TAE-20/1	1234	375	391	609	2588	328	547	453	703	406	406	922	391	297	3500	328	734	2688	390	610
TEA-20/10	813	78	94	2953	78	78	62	234	109	219	62	125	63	203	62	94	62	78	110	63
TJE-20/10	594	281	297	265	469	297	407	266	297	281	250	266	343	204	234	360	2266	484	265	250
TAE-20/10	1407	359	391	3218	547	375	469	500	406	500	312	391	406	407	296	454	2328	562	375	313
TEA-20/20	735	141	141	125	62	63	62	187	110	141	156	125	94	156	47	609	188	78	203	141
TJE-20/20	546	453	359	485	328	453	438	250	515	297	313	218	203	688	422	453	250	328	531	234
TAE-20/20	1281	594	500	610	390	516	500	437	625	438	469	343	297	844	469	1062	438	406	734	375

Figure 3. *TEA, TJE and TAE (see text) with 1, 10 and 20 second delay between requests to the* `CAP` *– with 20 clients making requests*



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
TEA-30/1	81	78	17	14	14	62	78	12	17	12	63	15	63	10	94	10	30	63	11	78	10	79	12	14	62	14	78	78	63	94
TJE-30/1	57	31	29	29	36	23	21	37	43	28	43	34	23	29	21	37	25	26	35	23	18	45	29	26	21	23	21	32	34	29
TAE-30/1	13	39	46	43	50	29	29	50	61	40	50	50	29	40	31	48	33	32	46	31	29	53	42	40	28	37	29	40	40	39
TEA-30/10	78	94	18	28	32	18	30	78	17	78	78	62	63	15	14	10	32	11	47	78	93	11	12	93	12	91	14	25	14	32
TJE-30/10	87	29	30	29	28	26	46	18	34	21	61	25	54	25	40	34	63	21	51	31	25	59	28	31	72	34	51	29	46	42
TAE-30/10	16	39	31	31	34	45	35	26	51	29	68	31	60	40	54	45	96	32	56	39	34	70	40	40	73	95	65	54	60	75
TEA-30/20	10	78	15	11	93	63	62	14	10	11	78	17	10	14	46	63	47	14	10	10	12	25	14	78	94	12	63	47	94	62
TJE-30/20	92	36	23	20	29	25	23	78	40	26	32	23	28	35	23	28	26	50	32	50	37	32	31	25	32	60	23	90	23	42
TAE-30/20	20	43	39	31	39	31	29	92	51	37	40	40	39	50	71	34	31	64	43	60	50	29	45	32	42	73	29	95	32	48

Figure 4. *TEA, TJE and TAE (see text) with 1, 10 and 20 second delay between requests to the* `CAP` *– with 30 clients making requests*

volved a data set obtained from the UCI machine learning repository [19], and as the data was already converted into the right format, the conversion time $T_c(ds)$ is zero. We can see that for all requests made to the `CAP`, this time is less than the time required to execute the service, equivalent to $2T_{md}(ds)+T_e$ in Section 2 – represented by TJE-20/* in the figure. This time represents an invocation made via the GT4.0 container to the J48 Web Service. Our results indicate the advantage of using the market mechanism for discovering suitable services, even with moderate sized data sets. Figure 4 demonstrates the execution with 30 clients – here we can see that our approach is scalable as the number of clients are increased. The peaks in the graph occur due to the workload (arising from execution of other Web Services) on the machines on which the J48 service is being hosted.

6. Conclusions

A Grid market for exchanging data mining services has been described. The market offers Catallactic ("free-market") market mechanisms for trading of basic services and resources. The data mining tasks are considered to be Complex services, which may have varying resource requirements during their execution. To identify Basic service and resource requirements, a data mining task was divided into specific phases. A key assumption in this work is the existence of a service rich environment – i.e. there will be a large number of service providers able to offer services for use by others. This is already beginning to happen, with the availability of public domain registry services – such as `xmethods.net`. The adoption of the "service-oriented" approach by the Grid community also indicates that this assumption is more widely shared.

To demonstrate the approach, an implementation of a Grid Catallactic market has been achieved, which consists of three layers: The CatCOVITE application, the Catallactic middleware layer for the negotiation of services and resources, and the base platform using GT4. The descriptions of the service and resource requirements are achieved through the use of WS-Agreement. Access to the market is provided through an access point (implemented as a Web Service), to prevent significant modifications to an existing application making use of the market.

Our approach integrates several properties important for "Open Systems", such as being able to work with partial knowledge and support for decentralization. The approach presented here is therefore general in scope, and may be applied in a number of different scenarios. The distinction between a Complex and Basic service also allows the existence of Complex Service providers (essentially, intermediaries who do not own any computational resources or services themselves, but primarily aggregate services provided by others). Such provides may dynamically modify Basic service instances based on price fluctuations in the market – thereby maintaining the price offered to some external user. This aspect of the work needs further investigation.

REFERENCES

1. O. Ardaiz, P. Chacn, I. Chao, F. Freitag, L. Navarro, "An Architecture for Incorporating Decentralized Economic Models in Application Layer Networks", Smart Grids Technologies Workshop, Utrecht, Holland, 2005.
2. O. Ardaiz, P. Artigas, T. Eymann, F. Freitag, L. Navarro, M. Reinicke, "The Catallaxy Approach for Decentralized Economic-based Allocation in Grid Resource and Service Markets", Special Issue on Agent-based Grid Computing, International Journal of Applied Intelligence, accepted.
3. R. Buyya, D. Abramson, J. Giddy, H. Stockinger, "Economic Models for Resource Management and Scheduling in Grid Computing", The Journal of Concurrency and Computation: Practice and Experience (CCPE), Wiley Press, May 2002.
4. CATNET Project deliverables, D3: Catallaxy Evaluation Report, March, 2003. Available at: `http://research.ac.upc.es/catnet/pubs/D3.pdf`.
5. R. Quinlan, "C4.5: Programs for Machine Learning", Morgan Kaufmann Publishers, San Mateo, CA, 1993
6. F.A. Hayek, W. Bartley, P. Klein, B. Caldwell, "The Collected Works of F. A. Hayek", University of Chicago Press, 1989.
7. Global Grid Forum (as on December 2005). Available at: `http://www.ggf.org/`.

8. Globus Toolkit version 4.0 (as on December 2005). Available at: http://www.globus.org
9. L. Joita, J.S. Pahwa, P. Burnap, A. Gray, O.F. Rana, J. Miles, "Supporting Collaborative Virtual Organisations in the Construction Industry via the Grid", Proceedings of the UK e-Science All Hands Meeting 2004, 31st Aug.-3rd Sept. 2004, Nottingham, UK, 2004.
10. J. Joseph, M. Ernest, C. Fellenstein, "Evolution of Grid Computing Architecture and Grid Adoption Models", IBM Systems Journal, Volume 43, Issue 4, January 2004.
11. Traversat, Abdelaziz, and Pouyoul, Project JXTA: "Loosely-Consistent DHT Rendezvous Walker", Sun Microsystems, Inc., http://www.jxta.org/project/www/docs/jxtadht.
12. Web Services Agreement Specification (WS-Agreement), 28 June 2005. Available via: https://forge.gridforum.org/.
13. WS-Resource Framework (as on December 2005), http://www.globus.org/wsrf/.
14. The WEKA Toolkit, University of Waikato, New Zealand. Available at: http://www.cs.waikato.ac.nz/ml/weka/. Last viewed: December 2005.
15. C. Zhu, Z. Liu, W. Zhang, W. Xiao, Z. Xu and D. Yang, "Decentralized Grid Resource Discovery Based on Resource Information Community", Journal of Grid Computing, (2), pp 261-277, 2004
16. A.S. Ali, O.F. Rana, I.J. Taylor, "Web Services Composition for Distributed Data Mining", ICPP 2005 Workshops, International Conference on Parallel Processing, 14-17 June 2005, pp. 11–18
17. Liviu Joita, Omer Rana, Pablo Chacin, Oscar Ardaiz, Isaac Chao, Felix Freitag, Leandro Navarro, "Application Deployment Using the Catallactic Grid Middleware", Middleware for Grid Computing at ACM/USENIX/IFIP Middleware 2005, Grenoble, France, November 2005
18. C. S. Yeo and R. Buyya, "A Taxonomy of Market-based Resource Management Systems for Utility-driven Cluster Computing", Technical Report, GRIDS-TR-2004-12, University of Melbourne, Australia, December 8, 2004
19. UCI Machine Learning Repository. Available at: http://www.ics.uci.edu/~mlearn/MLRepository.html. Last viewed: December 2005.

Appendix: Agreement Template and Offer

Note: The wsag: prefix has been removed from the tags in these descriptions.

```
<AgreementTemplateLite>
  <Name>DataMiningComplexService</Name>
  <Context>
    <AgreementInitiator>Cardiff-A
    </AgreementInitiator>
    <StartingTime>2005-12-12T13:00:00
    </StartingTime>
    <TerminationTime>2005-12-12T14:00:00
    </TerminationTime>
  </Context>
  <Terms>
    <BasicServiceType>DataMiningService
    </BasicServiceType>
    <NumberOfBasicServiceNodes>1 to 10
    </NumberOfBasicServiceNodes>
    <BasicServiceConstraints>
      <ResponseTimePerRequest>10
      </ResponseTimePerRequest>
    </BasicServiceConstraints>
    <PayForService>
    </PayForService>
  </Terms>
</AgreementTemplateLite>
```

Agreement Template

The AO is initiated by the agreement initiator (the MGS) in this case. An AO is as follows:

```
<AgreementOfferLite>
  <Name>DataMiningComplexService</Name>
  <Context>
    <AgreementInitiator>Cardiff-A
    </AgreementInitiator>
    <StartingTime>2005-12-12T13:00:00
    </StartingTime>
    <TerminationTime>2005-12-12T15:00:00
    </TerminationTime>
  </Context>
  <Terms>
    <BasicServiceType>DataMiningService
    </BasicServiceType>
    <NumberOfBasicServiceNodes>1
    </NumberOfBasicServiceNodes>
    <BasicServiceConstraints>
      <ResponseTimePerRequest>10
      </ResponseTimePerRequest>
    </BasicServiceConstraints>
    <PayForService>100
    </PayForService>
  </Terms>
</AgreementOfferLite>
```

Agreement Offer