SCHEDULING SOLUTION FOR GRID META-BROKERING USING THE PLIANT SYSTEM

József Dániel Dombi

Institute of Informatics, University of Szeged, Árpád tér 2., Szeged, Hungary

Attila Kertész

MTA SZTAKI, 1518 Budapest, P.O. Box 63, Hungary

Keywords: Pliant system, Sigmoid function, Grid computing, Meta-Brokering.

Abstract: In this paper we present some advanced scheduling techniques with a weighted fitness function for an adaptive Meta-Brokering Grid Service using the Pliant system, which enables a higher level utilization of the existing grid brokers. We construct and demonstrate the efficiency of our new algorithms in a grid simulation environment. The results given here demonstrate that the proposed novel scheduling technique produces better performance scores.

1 INTRODUCTION

In 1998, a new computing infrastructure called the Grid was born when the bible for the Grid (Kesselman and Foster, 1998) was published by Ian Foster et. al. Since then, Grid Computing has become an independent field of research; current Grid Systems are intented for numerous worldwide projects, and production Grids serve various user communities all around the world. The emerging Web technologies have influenced Grid development, and the latest solutions from related research fields (e.g. autonomous computing, artificial intelligence and peer-to-peer technologies) also need to be taken into account in order to achieve better resource utilization and successfully transform the currently separate production Grids to the future Internet of Services (Report, 2006).

As the management and advantageous utilization of highly dynamic, complex grid resources cannot be handled by the users themselves, various grid resource management tools have been developed and support different Grids. User requirements created certain properties that resource managers now provide. This development is continuing, and users still find it hard to distinguish brokers and to migrate their applications when they move to a different grid system. Scheduling in diverse and distributed environments requires sophisticated approaches because a

high uncertainty is present at several stages of a Grid environment. The main contribution of this paper lies in an enhanced scheduling solution based on the *Pliant System* (Dombi, 1997) that is applied to the resource management layer of grid middleware systems.

The Pliant system is very similar to a fuzzy system (Dombi, 1982). The difference between the two systems lies in the choice of operators. In fuzzy theory the membership function plays an important role, although the exact definition of this function is often not clear. In Pliant systems we use a so-called distending function, which represents a soft inequality. In the Pliant system the various operators (conjunction, disjunction and aggregation) are closely related to each other. We also use unary operators (negation and modifiers), which are also related to the Pliant system. In consequence, the Pliant system involves only those operators whose relationship is clearly defined and the whole system is based on these identities.

In the next section we shall introduce the Pliant system, operators and functions. In Section 3 we will describe meta-brokering in Grids, while in Section 4 we discuss algorithms of an adaptive scheduling technique that seek to provide better scheduling in global Grids. Section 5 describes an evaluation of our proposed solution and finally, in the last section we will

briefly summarize our results and draw some relevant conclusions.

2 COMPONENTS OF THE PLIANT SYSTEM

In fuzzy logic theory (Dombi, 1982) the membership function plays an important role. In pliant logic we introduce a distending function with a soft inequality. The *Pliant system* is a strict, monotonously increasing t-norm and t-conorm, and the following expression is valid for the generator function:

$$f_c(x)f_d(x) = 1, \qquad (1)$$

where $f_c(x)$ and $f_d(x)$ are the generator functions of the conjunctive and disjunctive logical operators. Here we use the representation theorem of the associative equation (Aczl, 1966):

$$f_c(x_1, x_2, \cdots, x_n) = f_c^{-1}\left(\sum_{i=1}^{n} f_c(x_i)\right) \qquad (2)$$

The generator function could be the *Dombi operator* (Dombi, 1982):

$$f(x) = \frac{1-x}{x}, f^{-1}(x) = \frac{1}{1+x} \qquad (3)$$

Besides the above-mentioned logical operators in fuzzy theory, there is also another non-logical operator. The reason for this that for real world applications it is not enough to use just conjunctive or disjunction operators (Zimmermann, 1991). The rational form of an aggregation operator is:

$$a_{\nu,\nu_0}(x_1, \cdots, x_n) = \frac{1}{1 + \frac{1-\nu_0}{\nu_0} \frac{\nu}{1-\nu} \prod_{i=1}^{n} \frac{1-x_i}{x_i}}, \qquad (4)$$

where ν is the neutral value and ν_0 is the threshold value of the corresponding negation.

The general form of the distending function is the following:

$$\delta_a^{(\lambda)}(x) = f^{-1}\left(e^{-\lambda(x-a)}\right), \lambda \in R, a \in R \qquad (5)$$

Here f is the generator function of the logical connectives, λ is responsible for the sharpness and a denotes the threshold value. The semantic meaning of $\delta_a^{(\lambda)}$ is

$$truth(a <_\lambda x) = \delta_a^{(\lambda)}(x) \qquad (6)$$

Important properties of the distending function are:

1. In the Pliant system f could be the generator function of the conjunctive operator or the disjunctive operator. The form of $\delta_a^{(\lambda)}(x)$ is the same in both cases.

2. In the Pliant concept the operators and membership are closely related.

2.1 Sigmoid Function

In the Dombi operator case, the distending function is the sigmoid function (see Figure 1):

$$\sigma_a^{(\lambda)}(x) = \frac{1}{1 + e^{-\lambda(x-a)}} \qquad (7)$$

Here, it is clear that:

1.
$$\sigma_a^{(\lambda)}(a) = \frac{1}{2} = \nu_0 \qquad (8)$$

2.
$$\left(\sigma_a^{(\lambda)}(a)\right)' = \frac{\lambda}{4} \qquad (9)$$

3.
$$\sigma_a^{(-\lambda)}(x) = n(\sigma_a^{(\lambda)}(x)) \text{ if } n(x) = 1 - x. \qquad (10)$$
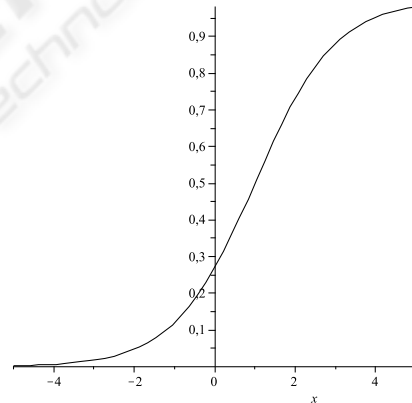

Figure 1: Sigmoid function.

The sigmoid function naturally maps the values to the (0,1) interval.

2.2 Kappa function

In order to make a decision we need to define a unary transformation for the values produced by the pliant operator. Using pliant logic, the general form of the modification operators is:

$$\kappa_\nu^\lambda = \frac{1}{1 + \left(\frac{\nu}{1-\nu} \frac{1-x}{x}\right)^\lambda} \qquad (11)$$

The behaviour of this function for different values of ν nand λ can be seen in figures 2 and 3.
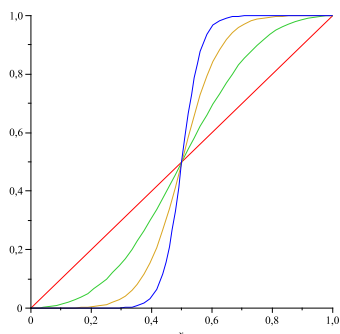

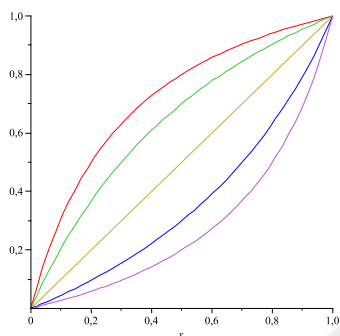Figure 2: Modification operator with the parameter values $\nu = 0.5, \lambda = 1, 2, 4, 8$.

Figure 3: Modification operator with the parameter values $\lambda = 1, \nu = 0.1, 0.3, 0.5, 0.7, 0.8$.

3 META-BROKERING IN GRID SYSTEMS

Meta-brokering refers to a higher level of resource management, which utilizes an existing resource or service brokers to access various resources. In some generalized way, it acts as a mediator between users or higher level tools and environment-specific resource managers. The main tasks of this component are: to *gather* static and dynamic broker properties, and to *schedule* user requests to lower level brokers, i.e. match job descriptions to broker properties. Finally the job needs to be *forwarded* to the selected broker.

Figure 4 provides a schematic diagram of the Meta-Broker (MB) architecture (Kertesz and Kacsuk, 2008), including the components needed to fulfil the above-mentioned tasks. Different brokers use different service or resource specification descriptions to understand the user request. These documents need to be written by the users to specify the different kinds of
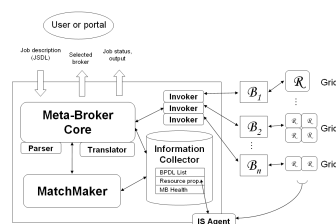

Figure 4: Components of the Meta-Broker.

service-related requirements. For the resource utilization in Grids, OGF (ogf, 1999) developed a resource specification language standard called JSDL (Anjomshoaa et al., 2005). As JSDL is general enough to describe the jobs and services of different grids and brokers, this is the default description format of MB. The *Translator* component of the Meta-Broker is responsible for translating the resource specification defined by the user to the language of the appropriate resource broker that MB selects to use for a given request. These brokers have various features for supporting different user needs, hence an extendable Broker Property Description Language (BPDL) (Kertesz and Kacsuk, 2008) is required to express metadata about brokers and the services they provide. The *Information Collector* (IC) component of MB stores data about the accessible brokers and historical data about previous submissions. This information tells us whether the chosen broker is available, and/or how reliable its services are. During broker utilization the successful submissions and failures are tracked, and for these events a ranking is updated for each special attribute in the BPDL of the appropriate broker (these attributes are listed above). In this way, the BPDL documents represent and store the dynamic states of the brokers. In order to support load balancing, there is an *IS Agent* (IS stands for Information System) reporting to the IC, which regularly checks the load of the underlying resources of each linked broker, and stores this data. The matchmaking process consists of the following steps: The *MatchMaker* (MM) compares the received descriptions to the BPDL of the registered brokers. This selection determines a group of brokers that can provide the required service. Otherwise, the request is rejected. In the second phase the MM counts a rank for each of the remaining brokers. This rank is calculated from the broker properties that the IS Agent updates regularly, and from the service completion rate that is updated in the BPDL for each broker. When all the ranks have been counted, the list of the brokers is ordered by these ranks. Finally the first broker of the priority list is selected, and the *Invoker* component forwards the request to the broker.

Regarding related works, other approaches try

to define common protocols and interfaces among scheduler instances enabling inter-grid usage. The meta-scheduling project in LA Grid (Rodero, 2008) aims to support grid applications with resources located and managed in different domains. They define broker instances with a set of functional modules. Each broker instance collects resource information from its neighbors and saves the information in its resource repository. The resource information is distributed in the different grid domains and each instance will have a view of all resources. The Koala grid scheduler (Iosup et al., 2007) was designed to work on DAS-2 interacting with Globus middleware services with the main features of data and processor co-allocation; lately it is being extended to support DAS-3 and Grid'5000. Their policy is to use a remote grid only if the local one is saturated. They use a so-called delegated matchmaking (DMM), where Koala instances delegate resource information in a peer-to-peer manner. Gridway introduces a Scheduling Architectures Taxonomy (Leal et al., 2009). Its Multiple Meta-Scheduler Layers use Gridway instances to communicate and interact through grid gateways. These instances can access resources belonging to different administrative domains. They also pass user requests to another domain, when the current one is overloaded. Comparing these related approaches, we can state that all of them use a new method to expand current grid resource management boundaries. Meta-brokering appears in a sense that different domains are being examined as a whole, but they rather delegate resource information among domains, broker instances or gateways through their own, implementation-dependent interfaces. Their scheduling policies focus on resource selection by usually aggregated resource information sharing, while our approach targets broker selection based on broker properties and performances.

4 SCHEDULING ALGORITHMS

In the previous sections we introduced the Pliant System and Grid Meta-Broker, and showed how the default matchmaking process is carried out. The main contribution of this paper is to *enhance* the scheduling part of this matchmaking process. To achieve this, we created a Decision Maker component based on functions of the *Pliant system*, and inserted it into the MatchMaker component of the Meta-Broker. The first part of the matchmaking is unchanged: the list of the available brokers is filtered according to the requirements of the actual job read from its JSDL. Then a list of the remaining brokers along with their perfor-

mance data and the background grid load are sent to the Decision Maker in order to determine the most suitable broker for the actual job. The scheduling techniques and the scheduling process are described below.

The Decision Maker uses a random number generator, and we chose a JAVA solution that generates pseudorandom numbers. The JAVA random generator class uses a uniform distribution and 48-bit seed and the latter is modified by a linear congruential formula (Knuth, 1997). We also developed a unique random number generator which generates random numbers with a given distribution. We call this algorithm the generator function. In our case we defined a score value for each broker, and we created the distribution based on the score value. For example, the broker which has the highest score number has the biggest chance of being chosen.

To improve the scheduling performance of the Meta-Broker we need to send the job to the broker that best fits the requirements, and executes the job without failures with the shortest execution time. Every broker has *four properties* that the algorithm can rely on: a success counter, a failure counter, a load counter and the running jobs counter.

- The success counter gives the number of jobs which had finished without any errors.
- The failure counter shows the number of failed jobs.
- The load counter indicates the actual load of the grid behind the broker (in percentage terms).
- The running jobs counter shows the number of jobs sent to the broker which have not yet finished.

We developed *two* different kinds of decision algorithms that take into account the above-mentioned broker properties. These algorithms define a score number for each broker and use the generator function to select a broker. Both algorithms use the kappa function to determine the broker's score number.

Because the Pliant system is defined in the $[0,1]$ interval, we need to *normalize* the input value. The two algorithms differ only in this step:

1. The first algorithm uses a linear transformation called Decision4.

2. The second algorithm uses the Sigmoid function to normalize the input values, which is called Decision5.

It is also important to *emphasize* that the closer the value is to one, the better the broker is, and if the value is close to zero, it means that the broker is not good. For example if the failure counter is high, both normalization algorithms should give a value close to

zero because it is not a good thing if the broker has a lot of failed jobs (see in Figure 5). The opposite of this case is true for the success counter (see Figure 6).
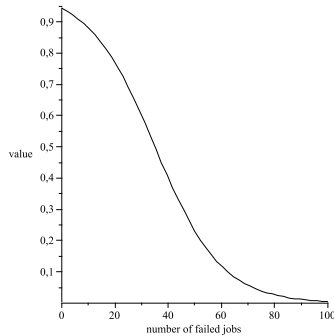

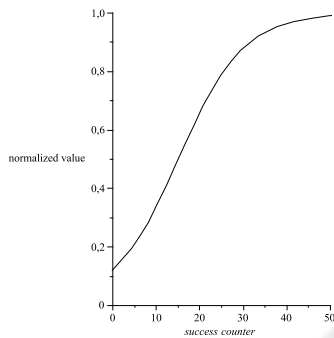Figure 5: Normalizing the failed jobs counter using Sigmoid function.

Figure 6: Normalizing the success counter using the Sigmoid function.

In the next step we can modify the normalized property value by using the same Kappa function (see Figure 7). We can also define the expected value of the normalization via the ν and λ parameters.
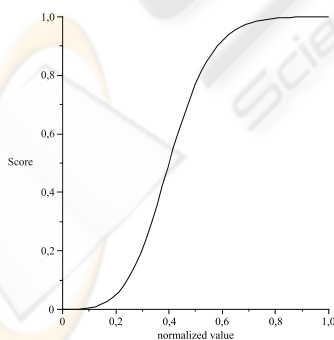

Figure 7: Normalized parameter values using the Kappa function.

To *calculate* the score value, we can make use of the conjunctive or aggregation operator. After running some tests we found that we get better results if

we use the aggregation operator. In this step the result is always a real number lying in the $[0,1]$ interval and then we multiply it by 100 to get the broker's score number.

When the Meta-Broker is running, the first two broker properties (the success and failure counters) are incremented via a feedback method that the simulator (or a user or portal in real world cases) calls after the job has finished. The third and fourth properties, the load value and the running jobs, are handled by the IS Agent of the Meta-Broker, queried from an information provider (Information System) of a Grid. During a simulation this data is saved to a database by the Broker entities of the simulator (described later and shown in Figure 8). This means that by the time we start the evaluation and before we receive feedback from finished jobs, the algorithms can only rely on the background load and running processes of the grids. To further enhance the scheduling we developed a *training process* that can be executed before the simulation in order to initialize the first and second properties. This process sends a small number of jobs with various properties to the brokers and sets the successful and failed jobs number at the BPDLs of the brokers. With this additional training method, we can expect shorter execution times because we will select more reliable brokers.

5 RESULTS

5.1 Evaluation

In order to evaluate our proposed scheduling solution, we created a general *grid simulation environment*, where all the related grid resource management entities could be simulated and coordinated. The GridSim toolkit (Buyya et al., 2002) is a fully extendable, widely used and accepted grid simulation tool. These are the main reasons why we chose this toolkit for our simulations. It can be used for evaluating VO-based resource allocation, workflow scheduling, and dynamic resource provisioning techniques in global grids. It supports the modeling and simulation of heterogeneous grid resources, users, applications, brokers and schedulers in a grid computing environment. It provides primitives for the creation of jobs (called gridlets), the mapping of these jobs to resources, and their management, thus resource schedulers can be simulated to study scheduling algorithms. GridSim provides a multilayered design architecture based on SimJava (Howell and Mcnab, 1998), a general purpose discrete-event simulation package implemented in Java. It is used for handling the interactions or

events among GridSim components. Each component in GridSim communicates with another via message passing operations defined by SimJava.

Figure 8: Meta-Brokering simulation environment based on GridSim.

Our general simulation architecture is shown in Figure 8. In the bottom right hand corner we can see the GridSim components used to simulate the grid system. The resources can be defined with different grid-types. These resources consist of more machines, for which workloads can be set. On top of this simulated grid infrastructure we can set up brokers. The Broker and Simulator entities were developed by us in order to simulate the meta-brokering process. Brokers are extended GridUser entities. Here

- they can be connected to one or more resources;

- different properties can be assigned to these brokers (agreement handling, co-allocation, advance reservation, etc.);

- some properties may be marked as unreliable;

- various scheduling policies can be defined (predefined ones: rnd – random resource selection, fcpu – resources having more free cpu time or fewer waiting jobs are selected, nfailed – resources having fewer machine failures are selected);

- in general resubmission is used when a job fails due to resource failure;

- after they report to the IS Grid load database by calling the feedback method of the Meta-Broker with the results of the job submissions (this database has a similar purpose to that of a grid Information System).

The Simulator is an extended GridSim entity. Hence

- it can generate a requested number of gridlets (jobs) with different properties, start and run times (length);

- it is connected to the created brokers and is able to submit jobs to them;

- the default job distribution is the random broker selection (where the middleware types should be taken into account);

- in the case of a job failure a different broker is selected for the actual job;

- it is also connected to the Grid Meta-Broker through its Web service interface and is able to call its matchmaking service for broker selection.

5.2 Evaluation Environment

Table 1 shows the *evaluation environment* used in our evaluation. The simulation setup was derived from real-life production grids: current grids and brokers support only a few special properties: here we used four. To determine the number of resources in our

Table 1: Evaluation environment setup.

Broker	Scheduling	Properties	Resources
1.	fcpu	A	6
2.	fcpu	A_F	8
3.	fcpu	A	12
4.	fcpu	B	10
5.	fcpu	B_F	10
6.	fcpu	B	12
7.	fcpu	B_F	12
8.	fcpu	C	4
9.	fcpu	C	4
10.	fcpu	$A_F D$	8
11.	fcpu	AD	10
12.	fcpu	AD_F	8
13.	fcpu	AB_F	6
14.	fcpu	ABC_F	10

simulated grids we compared the sizes of current production grids (EGEE VOs, DAS3, NGS, Grid5000, OSG, etc.). In the evaluation we utilized 14 brokers. We submitted 1000 jobs to the system, and measured the makespan of all the jobs. Out of the 1000 jobs 100 had no special properties, while for the rest of the jobs four key properties were distributed in the following way: 300 jobs had property A, 300 had B, 200 had C and 100 had D. The second column above denotes the scheduling policies used by the brokers: fcpu means the jobs are scheduled to the resource with the highest free cpu time. The third column shows the capabilities/properties (like coallocation, checkpointing) of the brokers: here we used A, B, C and D in the simulations. The F subscript means unreliability, a broker having the kind of property that may fail to execute a job with the requested service with a probablity of 0.5. The fourth column contains the number of resources utilized by a broker. As a background

workload, 50 jobs were submitted to each resource by the simulation workload entities during the evaluation timeframe. The SDSC BLUE workload logs were used for this purpose, taken from the Parallel Workloads Archive (PWA, 2009).

In order to test all the features of the algorithms, we submitted the jobs periodically: 1/3 of the jobs were submitted at the beginning then the simulator waited for 200 jobs to finish and update the performances of the brokers. After this phase the simulator again submitted 1/3 of all the jobs and waited for 200 more to finish. Lastly the remaining jobs (1/3 again) were submitted. In this way the broker performance results could be updated and monitored by the scheduling algorithms.
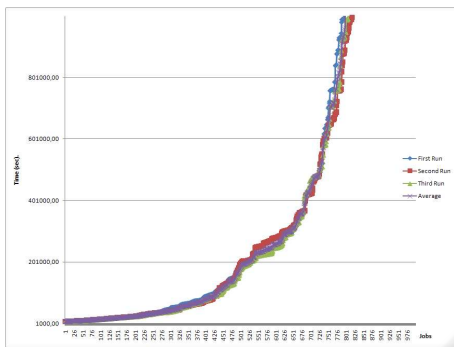
Figure 9: Results of Decision 4 algorithm.

In the previous section we explained how the two algorithms called Decision4 and Decision5 (both based on the Pliant system) work. For the evaluation part we repeated each experiment *three times*. The measured simulation results of the Decision4 algorithm can be seen in Figure 9. We noticed that the measured runtimes for the jobs were very close to each other. When comparing the various simulation types we always used the median: we counted the average runtime of the jobs in each of the three series and discarded the best and the worst simulations.

A comparison of the simulation results can be seen in Figure 10 above. In our previous paper (Kertesz et al., 2009) we used only random number generators to boost the Decision Maker, and proposed three algorithms called Decision1, Decision2 and Decision3. In that paper Decision3 gave the best results. This is why we will compare our new measurements with the results of this algorithm. We can see that for around 1/3 of the simulations, Decision3 provides better results, but the overall makespans are better for the new algorithms.

The simulation results for the algorithms with training can be seen in Figure 11. As we mentioned earlier, we used a training process to initiate the per-
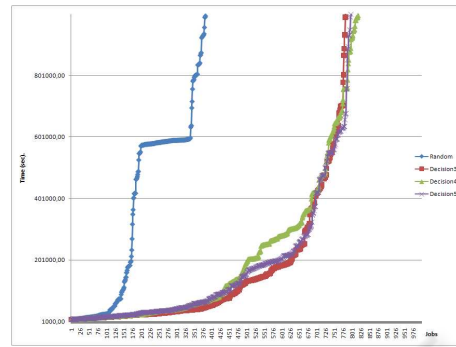

Figure 10: Simulation results for the three desicion algorithms compared with the random decision maker.
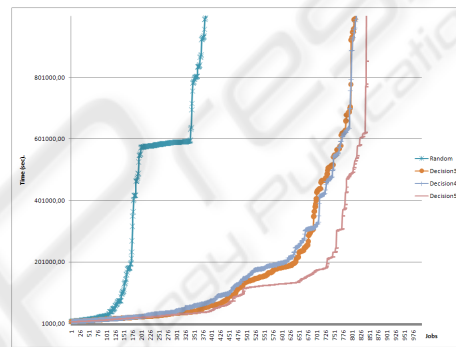

Figure 11: Simulation results for three desicion algorithms with training compared with the random decision maker.

formance values of the brokers before job submissions. In this way, the decisions for the first round of jobs can be made better. Upon examining the results, Decision4 still performs about the same as Decision3, but Decision5 clearly *overperforms* the other two.
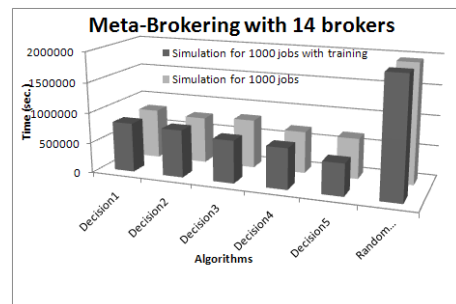

Figure 12: Simulation in the main evaluation environment.

In Figure 12 above we provide a graphical summary of the various evaluation phases. The columns show the average values of each evaluation run with the same parameter values. The results clearly show that the more intelligence (more sophisticated methods) we put into the system, the better the perfor-

mance. The *most advanced* version of our proposed meta-brokering solution is called the Decision Maker using the algorithm called *Decision5 with training*. Once the number of brokers and job properties are sufficiently high to set up this Grid Meta-Broker Service for inter-connecting several Grids, the new scheduling algorithms will be ready to serve thousands of users even under conditons of *high uncertainty*.

6 CONCLUSIONS

The Grid Meta-Broker itself is a standalone Web-Service that can serve both users and grid portals. The presented enhanced scheduling solution based on *Pliant functions* allows a higher level, interoperable brokering by utilizing existing resource brokers of different grid middleware. It gathers and utilizes meta-data about brokers from various grid systems to establish an adaptive meta-brokering service. We developed a new scheduling component for this Meta-Broker called Decision Maker that uses *Pliant functions* with a random generation in order to select a good performing broker for user jobs even under conditions of high uncertainty. We evaluated our algorithms in a grid simulation environment based on GridSim, and performed simulations with real workload samples. The evaluation results accord with our expected utilization gains: the enhanced scheduling provided by the revised Decision Maker results in a *more efficient* job execution.

ACKNOWLEDGEMENTS

REFERENCES

(1999). Open grid forum website. http://www.ogf.org.

(2009). Parallel workloads archive website. http://www.cs.huji.ac.il/labs/parallel/workload.

Aczl, J. (1966). *Lectures on Functional Equations and Applications*. Academic Press.

Anjomshoaa, A., Brisard, F., Drescher, M., Fellows, D., Ly, A., McGough, S., Pulsipher, D., and Savva, A. (2005). Job submission description language (jsdl) specification, version 1.0. Technical report. http://www.gridforum.org/documents/GFD.56.pdf.

Buyya, R., Murshed, M., and Abramson, D. (2002). Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. In *Journal of Concurrency and Computation: Practice and Experience (CCPE*, pages 1175–1220. Wiley Press.

Dombi, J. (1982). A general class of fuzzy operators, the de morgan class of fuzzy operators and fuzziness measures induced by fuzzy operators. *Fuzzy Sets and Systems*, 8.

Dombi, J. (1997). Pliant system. *IEEE International Conference on Intelligent Engineering System Proceedings, Budapest, Hungary*.

Howell, F. and Mcnab, R. (1998). Simjava: A discrete event simulation library for java. In *Proc. of the International Conference on Web-Based Modeling and Simulation*, pages 51–56.

Iosup, A., Epema, D.H.J., Tannenbaum, T., Farrellee, M., Livny, M. (2007). Inter-Operating Grids through Delegated MatchMaking. In *Proc. of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC07)*, Reno, Nevada.

Kertesz, A., Dombi, J.D., Dombi, J. (2009). Adaptive scheduling solution for grid meta-brokering. In *Acta Cybernetica*, Volume 19, pp. 105–123.

Kertesz, A. and Kacsuk, P. (2008). Meta-broker for future generation grids: A new approach for a high-level interoperable resource management. In *Grid Middleware and Services Challenges and Solutions*, pages 53–63. Springer US.

Kesselman, C. and Foster, I. (1998). *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers.

Knuth, D. E. (1997). *The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Leal, K., Huedo, E., Llorente, I.M. (2009). A decentralized model for scheduling independent tasks in Federated Grids. In *Future Generation Computer Systems*, Volume 25, Issue 8, pp. 840–852.

Rodero, I., Guim, F., Corbalan, J., Fong, L.L., Liu, Y.G., Sadjadi, S.M. (2008). Looking for an Evolution of Grid Scheduling: Meta-brokering. In *Proc. of Coregrid Workshop in Grid Middleware'07*, Dresden, Germany.

Report, N. G. G. (2006). Future for european grids: Grids and service oriented knowledge utilities – vision and research directions 2010 and beyond. Technical report. ftp://ftp.cordis.lu/pub/ist/docs/grids/ngg3_eg_final.pdf.

Zimmermann, H. (1991). *Fuzzy Set Theory and its applications*. Kluwer Academic, Dordrecht.