Improving the Dependability of Grids via Short-Term Failure Predictions

Artur Andrzejak and Demetrios Zeinalipour-Yazti and Marios D. Dikaiakos

Abstract Computational Grids like EGEE offer sufficient capacity for even most challenging large-scale computational experiments, thus becoming an indispensable tool for researchers in various fields. However, the utility of these infrastructures is severely hampered by its notoriously low reliability: a recent nine-month long study found that only 48% of jobs submitted in South-Eastern-Europe completed successfully. We attack this problem by means of proactive failure detection. Specifically, we attempt to predict site failures on short-term time scale by deploying machine learning algorithms to discover relationships between site performance variables and subsequent failures. Such predictions can be used by Resource Brokers for deciding where to submit new jobs, and help operators to take preventive measures. Our experimental evaluation on a 30-day trace from 197 EGEE queues shows that the accuracy of results is highly dependent on the selected queue, the type of failure, the preprocessing and the choice of input variables.

1 Introduction

Detecting and managing failures is an important step towards the goal of a dependable and reliable Grid. Currently, this is an extremely complex task that relies on over-provisioning of resources, ad-hoc monitoring and user intervention. Adapting ideas from other contexts such as cluster computing [11], Internet services [9, 10] and software systems [12] is intrinsically difficult due to the unique characteristics of Grid environments. Firstly, a Grid system is not administered cen-

A. Andrzejak
Name, Zuse Institute Berlin (ZIB), Takustraße 7, 14195 Berlin, Germany, e-mail: andrzejak@zib.de

Demetrios Zeinalipour-Yazti and Marios D. Dikaiakos
Department of Computer Science, University of Cyprus, CY-1678, Nicosia, Cyprus e-mail: {dzeina,mdd}@cs.ucy.ac.cy

trally; thus it is hard to access the remote sites in order to monitor failures. Moreover failure feedback mechanisms cannot be encapsulated in the application logic of each individual Grid software, as the Grid is an amalgam of pre-existing software libraries, services and components with no centralized control. Secondly, these systems are extremely large; thus, it is difficult to acquire and analyze failure feedback at a fine granularity. Lastly, identifying the overall state of the system and excluding the sites with the highest potential for causing failures from the job scheduling process, can be much more efficient than identifying many individual failures.

In this work, we define the concept of *Grid Tomography*[1] in order to discover relationships between Grid site performance variables and subsequent failures. In particular, assuming a set of monitoring sources (system statistics, representative low-level measurements, results of availability tests etc.) that characterize Grid sites we predict with high accuracy site failures on short-term time scale by deploying various off-the-shelf machine learning algorithms. Such predictions can be used for deciding where to submit new jobs and help operators to take preventive measures.

Through this study we manage to answer several questions that have to our knowledge not been addressed before. Particularly, we provide answers to questions such as: *"How many monitoring sources are necessary to yield a high accuracy?"'*; *"Which of them provide the highest predictive information?"*, and *"How accurately can we predict the failure of a given Grid site X minutes ahead of time?"* Our findings support the argument that Grid tomography data is indeed an indispensable resource for failure prediction and management. Our experimental evaluation on a 30-day trace from 197 EGEE queues shows that the accuracy of results is highly dependent on the selected queue, the type of failure, the preprocessing and the choice of input variables.

This paper builds upon on previous work in [20], in which we presented the preliminary design of FailRank architecture. In FailRank, monitoring data is continuously coalesced into a representative array of numeric vectors, the *FailShot Matrix (FSM)*. FSM is then continuously ranked in order to identify the K sites with the highest potential to feature some failure. This allows the Resource Broker to automatically exclude the respective sites from the job scheduling process. FailRank is an architecture for on-line failure ranking using linear models, while this work investigates the problem of predicting failures by deploying more advanced, non-linear *classification algorithms* from the domain of machine learning.

In summary, this paper makes the following contributions:

- We propose techniques to predict site failures on short-term time scale by deploying machine learning algorithms to discover relationships between site performance variables and subsequent failures;
- We analyze which sources of monitoring data have the highest predictive information and determine the influence of preprocessing and prediction parameters on the accuracy of results;

[1] *Grid Tomography* refers in our context to the process of capturing the state of a grid system by sections, i.e., individual state attributes, (*tomos* is the Greek word for *section*.)

- We experimentally validate the efficiency of our propositions with an extensive experimental study that utilizes a 30-day trace of Grid tomography data that we acquired from the EGEE infrastructure.

The remainder of the paper is organized as follows: Section 2 formalizes our discussion by introducing the terminology. It also describes the data utilized in this paper, its preprocessing, and the prediction algorithms. Section 3 presents an extensive experimental evaluation of our findings obtained by using machine learning techniques. Finally, Section 4 concludes the paper.

2 Analyzing Grid Tomography Data

This section starts out by overviewing the anatomy of the EGEE Grid infrastructure and introducing our notation and terminology. We then discuss the tomography data utilized in our study, and continue with the discussion of pre-processing and modeling steps used in the prediction process.

2.1 The Anatomy of a Grid

A Grid interconnects a number of remote clusters, or *sites*. Each site features heterogeneous resources (hardware and software) and the sites are interconnected over an open network such as the Internet. They contribute different capabilities and capacities to the Grid infrastructure. In particular, each site features one or more *Worker Nodes*, which are usually rack-mounted PCs. The *Computing Element* runs various services responsible for authenticating users, accepting jobs, performing resource management and job scheduling. Additionally, each site might feature a *Local Storage* site, on which temporary computation results can reside, and local *Software* libraries, that can be utilized by executing processes. For instance, a computation site supporting mathematical operations might feature locally the *Linear Algebra PACKage (LAPACK)*. The Grid middleware is the component that glues together local resources and services and exposes high-level programming and communication functionalities to application programmers and end-users. EGEE uses the gLite middleware [6], while NSF's TeraGrid is based on the Globus Toolkit [5].

2.2 The FailBase repository

Our study uses data from our *FailBase Repository* which characterizes the EGEE Grid in respect to failures between 16/3/2007 and 17/4/2007 [14]. FailBase paves the way for the community to systematically uncover new, previously unknown patterns and rules between the multitudes of parameters that can contribute to failures

in a Grid environment. This database maintains information for 2,565 Computing Element (CE) *queues* which are essentially sites accepting computing jobs. For our study we use only a subset of queues for which we had the largest number of available types of monitoring data. For each of them the data can be thought of as a *time-series*, i.e., a sequence of pairs (timestamp,value-vector). Each value-vector consists of 40 values called *attributes*, which correspond to various sensors and functional tests. That comprises the *FailShot Matrix* that encapsulates the Grid failure values for each Grid site for a particular timestamp.

2.3 Types of monitoring data

The attributes are subdivided into four groups A, B, C and D depending of their source as follows [13]:

A. *Information Index Queries (BDII)*: These 11 attributes have been derived from LDAP queries on the Information Index hosted on *bdii101.grid.ucy.ac.cy*. This yielded metrics such as the number of free CPUs and the maximum number of running and waiting jobs for each respective CE-queue.
B. *Grid Statistics (GStat)*: The raw basis for this group is data downloaded from the monitoring web site of Academia Sinica [7]. The obtained 13 attributes contain information such as the geographical region of a Resource Center, the available storage space on the Storage Element used by a particular CE, and results from various tests concerning BDII hosts.
C. *Network Statistics (SmokePing)*: The two attributes in this group have been derived from a snapshot of the *gPing* database from ICS-FORTH (Greece). The database contains network monitoring data for all the EGEE sites. From this collection we measured the average round-trip-time (RTT) and the packet loss rate relevant to each South East Europe CE.
D. *Service Availability Monitoring (SAM)*: These 14 attributes contain information such as the version number of the middleware running on the CE, results of various replica manager tests and results from test job submissions. They have been obtained by downloading raw html from the CE sites and processing them with scripts [4].

The above attributes have different significance when indicating a site failure. As group D contains functional and job submission tests, attributes in this group are particularly useful in this respect. Following the results in Section 3.2 we regard two of these `sam` attributes, namely `sam-js` and `sam-rgma` as failure indicators. In other words, in this work we regard certain values of these two attributes as queue failures, and focus on predicting their values.

2.4 Preprocessing

The preprocessing of the above data involves several initial steps such as masking missing values, (time-based) resampling, discretization, and others (these steps are not a part of this study, see [13, 14]). It is worth mentioning that data in each group has been collected with different frequencies (A, C: once a minute, B: every 10 minutes, D: every 30-60 minutes) and resampled to obtain a homogeneous 1-minute sampling period. For the purpose of this study we have further simplified the data as follows: all missing or outdated values have been set to -1, and we did not make difference in severity of errors. Consequently, in our attribute data we use -1 for "invalid" values, 0 to indicate normal state, and 1 to indicate a faulty state. We call such a modified vector of (raw and derived) values a *sample.*

In the last step of the preprocessing, a sample corresponding to time T is assigned a *(true) label* indicating a future failure as follows. Having decided which of the `sam` attributes S represents a failure indicator, we set this label to 1 if any of the values of S in the interval $[T+1, T+p]$ is 1; otherwise the label of the sample is set to 0. The parameter p is called the *lead time*. In other words, the label indicates a future failure if the `sam` attribute S takes a fault-indicating value at *any* time during the subsequent p minutes.

2.5 Modeling methodology

Our prediction methods are *model-based*. A *model* in this sense is a function mapping a set of raw and/or preprocessed sensor values to an output, in our case a binary value indicating whether the queue is expected to be healthy (0) or not (1) in a specified future time interval. While such models can take a form of a custom formula or an algorithm created by an expert, we use in this work a *measurement-based* model [17]. In this approach, models are extrapolated *automatically* from historical relationships between sensor values and the simulated model output (computed from offline data). One of the most popular and powerful class of the measurement-based models are *classification algorithms* or *classifiers* [19, 3]. They are usually most appropriate if outputs are discrete [17]. Moreover, they allow the incorporation of multiple inputs or even functions of data suitable to expose its information content in a better way than the raw data. Both conditions apply in our setting.

A classifier is a function which maps a d-dimensional vector of real or discrete values called *attributes* (or *features)* to a discrete value called *class label*. In the context of this paper each such vector is a sample and a class label corresponds to the true label as defined in Section 2.4. Note that for an error-free classifier the values of class labels and true labels would be identical for each sample. Prior to its usage as a predictive model, a classifier is *trained* on a set of pairs (sample, true label). In our case samples have consecutive timestamps. We call these pairs the *training data* and denote by D the maximum amount of samples used to this purpose.

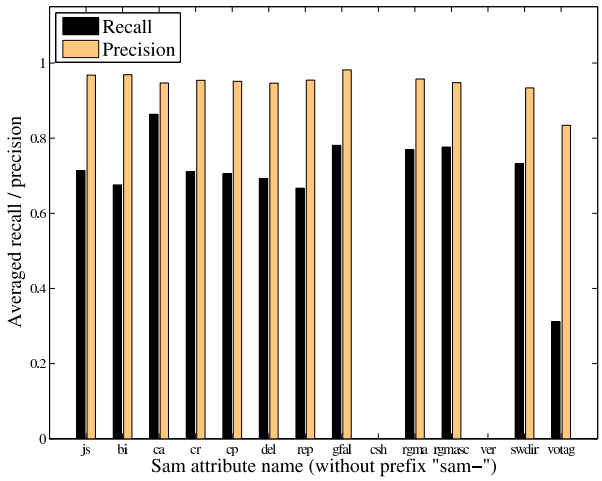

Fig. 1 Recall and Precision of each `sam` attribute

A trained classifier is used as a predictive model by letting it compute the class label values for a sequence of samples following the training data. We call these samples *test data*. By comparing the values of the computed class labels against the corresponding true labels we can estimate the accuracy of the classifier. We also perform model updates after all samples from the test data have been tested. This number - expressed in minutes or number of samples - is called the *update time*.

In this work we have tested several alternative classifiers such as C4.5, LS, Stumps, AdaBoost and Naive Bayes. The interested reader is referred to [3, 16] for a full description of these algorithms.

3 Experimental Results

Each prediction run (also called *experiment*) has a controlled set of preprocessing parameters. If not stated otherwise, the following default values of these parameters are used. The size of the training data D is set to 15 days or 21600 samples, while the model update time is fixed to 10 days (14400 samples). We use a lead time of 15 minutes. The input data groups are A and D, i.e., each sample consists of $11 + 14$ attributes from both groups. On this data we performed attribute selection via the backward branch-and-bound algorithm [16] to find 3 best attributes used as the classifier input. As classification algorithm we deployed the C4.5 decision tree algorithm from [15] with the default parameter values.

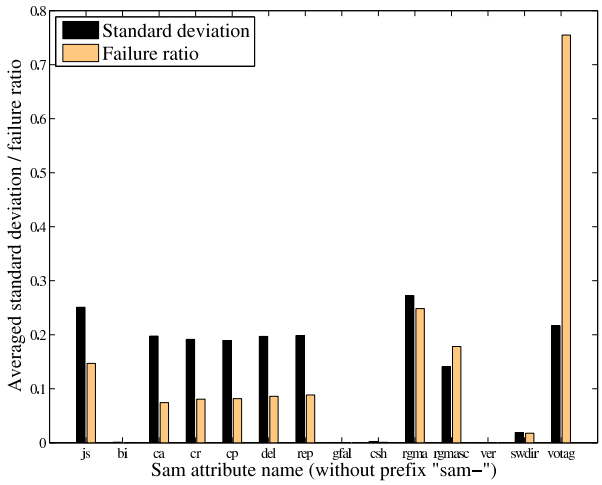

Fig. 2 Standard deviation and failure ratio for each `sam` attribute

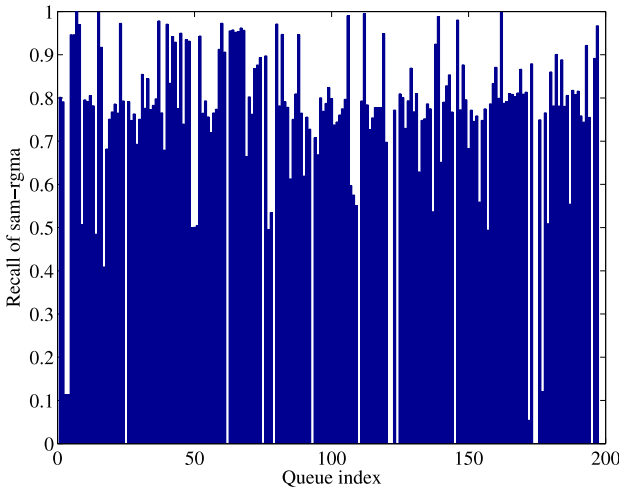

Fig. 3 Recall of attribute `sam-rgma` for all 197 queues

3.1 Evaluation metrics: recall and precision

During preprocessing, each training or test sample is assigned a *true label*: a value of 1 indicates a failure at the corresponding sample time, and a value 0 indicates no failure. During testing, a classifier assigns to each test sample a *predicted label* with analogous values. Obviously, the more frequently both values agree, the higher the quality of predictions. For the purpose of failure prediction cases with true label

equal to 1 are especially interesting. This gives rise to the following definitions common in the field of document retrieval.

For all test examples in a single experiment, *recall* is the number of examples with both predicted and true label equal 1 divided by number of cases with true label equal 1. This metrics estimates the probability that a failure is indeed predicted. The *precision* is the ratio of the number of examples with both predicted and true labels equal 1 to the number of examples with predicted label equal 1. It is interpreted as the probability that a predicted failure really occurs. We use in the following these two metrics to evaluate prediction accuracy.

3.2 Analysis of prediction accuracy

We shall next present an extensive experimental study, which focuses on two aspects: First, we investigate the influence of monitoring data groups as well as various preprocessing and mining parameters on the accuracy of results. Second, we seek to determine the highest prediction accuracy (measured in terms of recall and precision) that can be achieved depending on specific requirements on the predictions. For example, one type of the latter questions is: *how accurately can we predict the behavior of a Grid site X minutes ahead of time?*

Selecting the target attributes.

First we study which `sam` attributes are most interesting in terms of prediction accuracy and variance. We compute recall and precision for each combination of queue / `sam` attribute. Figure 1 shows these results for each particular `sam` attribute averaged over all queues. The preliminary conclusion from the figure is that most of the `sam` attributes (i.e., 12 out of the 14) are good choices for yielding a high recall/precision.

Consequently, we also considered the *failure ratio*: the ratio of all samples indicating a failure (in respect to the chosen target attribute) to all samples. Figure 2 shows these values for each `sam` attribute, averaged over all queues. The attributes `sam-bi, sam-gfal, sam-csh, sam-ver` and `sam-swdir` had a low failure ratio and standard deviation and were consequently excluded from further consideration.

We additionally ranked the remaining attributes according to their importance and their recall values, and consequently decided to only focus on the following two attributes:

- **sam-js**: This is a test that submits a simple job for execution to the Grid and then seeks to retrieve that job's output from the UI. The test succeeds only if the job finishes successfully and the output is retrieved.
- **sam-rgma**: R-GMA [2] is the Relational Grid Monitoring Architecture which makes all Grid monitoring data appear like one large Relational Database that

may be queried in order to find the information required. The `sam-rgma` test tries to insert a tuple and run a query for that tuple. The test returns success if all operations are successful.

Figure 3 shows that the recall of `sam-rgma` varies strongly among the queues. We observed a similar behavior for the failure indicator `sam-js` but omit these results for brevity.

Data characteristics and accuracy.

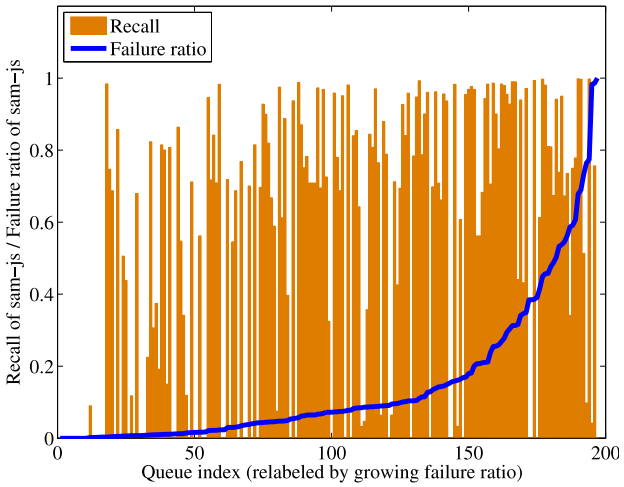

Fig. 4 Recall vs. sorted failure ratio of `sam-js` for all 197 queues

Next, we investigated the key characteristics of the data and how their variations influence the prediction accuracy. For each of the 197 queues and for the two target attributes (`sam-js` and `sam-rgma`) we computed the failure ratio as defined above. We then sorted all queues by increasing failure ratios and plotted the corresponding recall values for predictions with standard values. As seen in Figure 4 there is obviously no relationship between failure ratio and prediction accuracy. The same conclusions apply for the `sam-rgma` attribute.

We have also inspected visually the failure patterns over time in our data. Typically, an occurrence of a failure or non-failure is followed by a large number of samples of the same kind, i.e., the failure state does not change frequently; see top graph in Figure 5. Also typically the prediction errors occur right after the change in the failure state. This indicates that the value of the last historical sample of the target attribute was a good indicator of its future value.

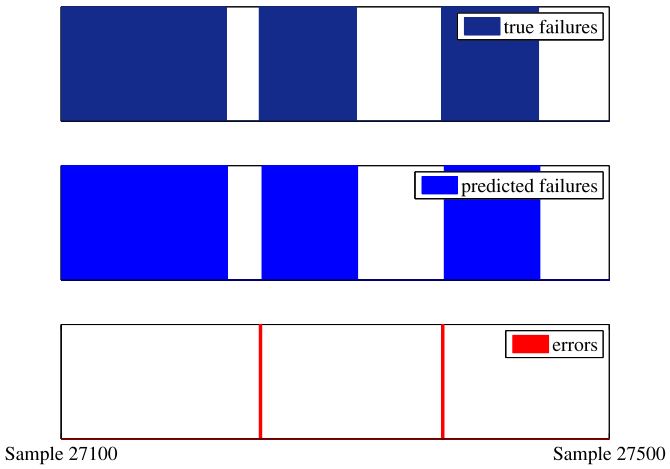

Fig. 5 Comparison of true and predicted failures for a typical interval of data (queue number 6, attribute `sam-js`, samples 27100 to 27500)

Effects of different classification algorithms.

Despite the theoretical knowledge and practical evidence that no classification algorithm can perform significantly better than others [8, 1] we experienced substantial deviations in recall and precision values for different algorithms, in the absence of attribute selection. We attribute this to the potentially high dimensionality of the input data (up to 40 attributes if all input groups are used) and a relatively large noise in the data. Figure 6 shows the recall values of five classification algorithms (see [15]) for the attribute sam-js averaged over 10 randomly selected queues (indexes 6, 9, 19, 54, 62, 75, 86, 137, 163, 188) without and with an attribute selection algorithm. Other algorithms such as k-nearest neighbor classifier or Support Vector Machine did not produce representative results due to memory or implementation problems in the used libraries [16, 15]. Figure 6 tells us that the AdaBoost algorithm (combined with Stumps) yielded best recall values. Furthermore, attribute selection improved the accuracy in all cases but for C4.5. Despite of this fact, C4.5 has been used as it had very small running time compared e.g. to AdaBoost.

4 Conclusions

In this paper we attack the problem of low reliability in job completion of Grid systems by means of proactive failure detection. Specifically, we predict site failures on short-term time scale by deploying classification algorithms that discover the relationships between site performance variables and subsequent failures. Our

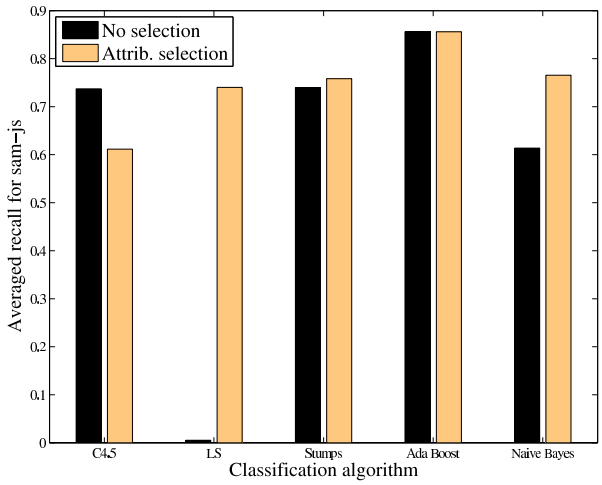

Fig. 6 Recall (`sam-js`) for five different classification algorithms without and with attribute selection (averaged over 10 queues)

experimental evaluation on a 30-day trace from 197 EGEE queues shows that the accuracy of results can be significantly high in many cases.

Acknowledgements

This work was supported in part by the European Union under projects CoreGRID (# IST-2002-004265) and EGEE (#IST-2003-508833). The authors would like to thank Kyriacos Neocleous and Yannis Ioannou from the University of Cyprus for their valuable help in constructing the Failbase repository and Charalampos Gkikas from ICS/FORTH for providing access to

References

1. A. Andrzejak and L. Silva. Using machine learning for non-intrusive modeling and prediction of software aging. In *IEEE/IFIP Network Operations & Management Symposium (NOMS 2008)*, Salvador de Bahia, Brazil, Apr 7–11 2008.
2. A. Cooke et. al. The Relational Grid Monitoring Architecture: Mediating Information about the Grid. *Journal of Grid Computing*, 2(4):323–339, 2004.
3. R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley and Sons, 2001. 0-471-05669-3.
4. EGEE. Service availability monitoring (SAM), http://sam-docs.web.cern.ch/sam-docs/.
5. I. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. *Journal of Computer Science and Technology*, 21(4):513–520, 2006.

6. Glite. Glite middleware, http://glite.org/.
7. GStat. Grid statistics (gstat), http://goc.grid.sinica.edu.tw/gstat/.
8. E. J. Keogh, S. Lonardi, and C. A. Ratanamahatana. Towards parameter-free data mining. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 206–215, August 2004.
9. E. Kiciman and A. Fox. Detecting application-level failures in component-based internet services, June 2004.
10. E. Kiciman and L. Subramanian. Root cause localization in large scale systems. In *In Proceedings of the 1 st Workshop on Hot Topics in System Dependability (HotDep-05*. IEEE Computer Society, June 2005.
11. S. Krishnamurthy, W. H. Sanders, and M. Cukier. A dynamic replica selection algorithm for tolerating timing faults. In *2001 International Conference on Dependable Systems and Networks (DSN 2001) (formerly: FTCS)*, pages 107–116, Goteborg, Sweden, July 2001. IEEE Computer Society.
12. M. E. Locasto, S. Sidiroglou, and A. D. Keromytis. Application communities: Using monoculture for dependability. In *In Proceedings of the 1 st Workshop on Hot Topics in System Dependability (HotDep-05*, pages 288–292, 2005.
13. K. Neocleous. Failure analysis, prediction and management on the EGEE grid infrastructure. Master's thesis, University of Cyprus, August 2007.
14. K. Neocleous, M. D. Dikaiakos, P. Fragopoulou, and E. Markatos. Failure management in grids: The case of the EGEE infrastructure. *Parallel Processing Letters*, 17(4):391–410, Dec. 2007.
15. D. G. Stork, E. Yom-Tov, and R. O. Duda. *Computer manual in MATLAB to accompany Pattern Classification*. Wiley, second edition, 2004.
16. F. van der Heijden, R. P. W. Duin, D. de Ridder, and D. M. J. Tax. *Classification, Parameter Estimation and State Estimation*. John Wiley & Sons, 2004.
17. R. Vilalta, C. V. Apte, J. L. Hellerstein, S. Ma, and S. M. Weiss. Predictive algorithms in the management of computer systems. *IBM Systems Journal*, 41(3):461–474, 2002.
18. WISDOM. Initiative for grid-enabled drug discovery against neglected and emergent diseases, http://wisdom.eu-egee.fr.
19. I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition edition, 2005.
20. D. Zeinalipour-Yazti, H. Papadakis, C. Georgiou, and M. Dikaiakos. Metadata ranking and pruning for failure detection in grids. *Parallel Processing Letters*, 18(3):371–390, Sept. 2008.