RJ10372 (A0602-044) February 22, 2006
Computer Science

IBM Research Report

ACE: Classification for Information Lifecycle Management

Gauri Shah, Kaladhar Voruganti
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

Piyush Shivam
Department of Computer Science
Duke University
Durham, NC 27708

Maria Alvarez
Department of Computer Science
University of California
Santa Barbara, CA 93106

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

ACE: Classification for Information Lifecycle Management

Gauri Shah, Kaladhar Voruganti
IBM Almaden Research Center
650 Harry Road
San Jose, CA 95120
gauris, kaladhar@us.ibm.com

Piyush Shivam*
Dept. of CS
Duke University
Durham, NC 27708
shivam@cs.duke.edu

Maria Alvarez*
Dept. of CS
University of California
Santa Barbara, CA 93106
malvarez@cs.ucsb.edu

Abstract

One of the principal problems in Information Lifecycle Management is to align the business value of data with the most cost-effective and appropriate storage infrastructure. In this paper, we introduce ACE: a framework of tools for ILM, that classifies data and storage resources, and generates a data placement plan for informed utilization of the available storage resources in the system. The goal of ACE is to design a data placement plan that provides cost benefits to an organization while allowing efficient access to all important data. To achieve this goal, ACE uses a policy-based approach to classify data and storage based on the metadata attributes and capabilities respectively. The main advantage of using ACE is that it enables appropriate usage of under-utilized storage systems without extensive human intervention. Another key characteristic of ACE is that it uses a policy-based architecture to automate the process of data valuation and storage classification.

We implement the ACE framework and evaluate its benefits for three real data sets. One data sets consists of 1.28 *million anonymous medical industry record files of total size* 1461*GB, and we show that ACE provides a cost benefit of greater than* 70% *over the lifetime of the data. In addition to the novel valuation algorithms and overall architecture, we also demonstrate optimizations that reduce the total performance time to* 85% *of the time taken without these optimizations, while still maintaining classification accuracy of over* 85%*.*

1 Introduction

Storage needs of most enterprises are increasing at an exponential pace because organizations are automating more and more of their business processes, which is in turn leading to more data getting digitized and stored persistently. A study conducted by IBM predicted that the 3.2 million exabytes of information that existed on earth in the year 2001 would reach 43 million exabytes by the year 2005 [3]. This large corpus of data poses many challenges for intelligent data and storage management. In addition, new legislation such as Sarbannes-Oxley, HIPAA, Basle II, and IFRS [26, 15, 14], which require legal compliance of the storage of data adds to the complexity of data management.

Organizations can only continue to increase the size of their storage facilities or find intelligent ways to reduce the overall size and number of stored files. Most organizations are deploying storage area network (SAN) controllers, and network attached storage (NAS) servers at a fast rate to satisfy their organization's growing storage demand. IDC reports that the amount of new storage capacity installed each year is increasing by almost 80% annually [19]. Gartner also reports that an enterprise spends an average of three dollars managing storage for every one dollar spent on storage hardware [11].

However, the percentage of useful data residing on these expensive storage systems is becoming a very small percentage of the overall storage space being utilized. Data with lower business value such as temporary data, infrequently-accessed data and orphaned data (data cannot be traced to any application) is residing on expensive storage media. Thus, there is a need to manage both data and storage resources over time so that the storage resources of an enterprise can be optimally utilized. Further, as the total number of IT workers is increasing approximately at only 5% per year [19], there is a strong requirement to automate this management of resources.

SNIA uses the term *Information Lifecycle Management (ILM)* to addresses this issue. Content management, hierarchical storage management, and storage resource management, e.g., [10, 23, 25] are some of the tools that are trying to solve this problem under the umbrella of ILM. These tools provide mechanisms that allow administrators to classify their data during the deployment of a new application's storage. However, these tools are limited in their functionality in providing optimal storage resource utilization in existing system environments. In this paper, we present ACE, a new framework of tools for ILM, that transforms under-utilized legacy storage sys-

[1]This work was done when the author was a summer intern at IBM Almaden Research Center

tems into ones where the right type of data resides on the right type of storage at the right time.

Transforming legacy storage infrastructures into business-value-aware infrastructures is a time-consuming and difficult process due to the following reasons:

Time Consuming Process: It takes system administrators weeks (if not months) to carefully classify and assign the correct business value to all of the existing data in data stores. System administrators not only have to deal with a large volume of data but also with the heterogeneous aspects of the data stores.

Lack of Application-Data Relationships: It is time consuming to determine which data belongs to which application. For many applications, this information is not easily available making it difficult to value the data based on the application that uses it.

Temporal nature of business value: The business value of data does *not* remain constant [8]. For example, tax-related data has high business value during the tax season, and relatively lower business value otherwise. Thus, it is important for the storage system to dynamically migrate data between different classes of storage according to its changing business value.

Non-triviality of Data Valuation: Some administrators are capable of assigning business value to their data, but most of them need guidance in this task. Administrators have to be aware that different metadata attributes are relevant for different types of data classification.

ACE is a new framework of tools for ILM that addresses the above problems. The ACE framework consists of a data classification engine, a storage classification engine, and a data placement engine that maps the data to the appropriate storage. The goal of ACE is to help system administrators classify an organization's data and storage resources appropriately so that the data is placed on a matching class of storage resources according to its business value. The key features of ACE are as follows:

Provides Classification and Data Placement: ACE semi-automates the processes of determining the business value of the data, identifying different classes of data based on their business values, identifying the different tiers of storage quality, and aligning the data with the right quality of storage.

Uses Policy-driven Business Valuation: In order to aid administrators to specify the business value of data, ACE has a policy-driven valuation mechanism. The policies determine how the data gets mapped to different business values, and the storage gets mapped to different tiers of storage quality. This policy specification is very flexible and customizable.

Handles Temporal Business Value: ACE's data classification and data placement engines are capable of handling the *temporal* nature of the data business values by monitoring the system and the changing metadata characteristics.

Optimizes Performance: ACE uses several novel algorithms that improve the classification performance by reducing the domain space of the data that needs to be processed as well as the policies used for classification.

In addition to developing the novel algorithms for classification and the entire ACE framework, we also implement ACE and evaluate its performance in a real deployment on three real data sets. Two of the data sets are local code files and personal user data. The third set represents a 1.4TB of anonymous medical record files. We evaluate the cost benefits of using ACE for data placement, and show that ACE reduces the total storage cost by upto 70% over the lifetime of the data. We demonstrate that our novel algorithms, e.g. example-based classification which uses linear regression techniques, give improved performance and flexibility of policy specification with little classification error. We also prove that our performance optimizations such as file sampling in the data domain reduce the total performance time by almost 85% of the time taken without these optimizations, with only 15% classification errors.

The rest of the paper is organized as follows. Section 2 describes the ACE architecture. This section includes the details of the classification algorithms and the classification performance enhancement techniques. Section 3 describes the details of our evaluation framework. Section 4 discusses the related work. Finally, our conclusions and future work are presented in Section 5.

2 ACE Architecture

ACE provides a modular framework to scan an existing system for its data and storage resources, and gives a data placement solution to best utilize the available storage resources. ACE provides cost benefits to an organization by mapping appropriate data to enterprise storage and thus reducing the amount of such storage required. The objectives of ACE are three-fold:

1. To provide a *business valuation* to the data based on mining of their metadata attributes.
2. To determine the different tiers of storage quality available.
3. To provide a suitable *data placement* to ensure informed use of the storage resources.

Figure 1 shows the high-level ACE architecture.

In order to achieve its objectives, the ACE framework has the following three main components:

Data Classification Engine: This component is responsible for mining the metadata attributes of the data,

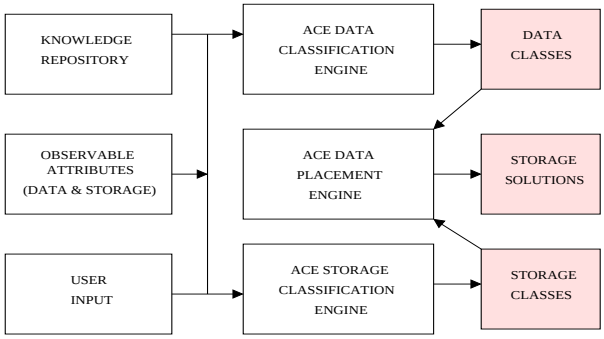


Figure 1: ACE architecture

and providing an appropriate business value to the data based on the policies. Different kinds of policies ares used as explained in Section 2.2. The output of this engine is a set of data classes each of which comprises of a collection of data objects which have similar characteristics and the same business values. Thus, there is a 1:1 mapping between the valuation provided to the data and the data classes generated using these values. We note that we only use metadata attributes for data classification and do not deal with content classification. However, as we point out later on in Section 2.3, the ACE architecture is complementary to, and flexible enough to accommodate content-based classification as well.

In our evaluation, we use the range of $[1-9]$ to assign business values to the data, 9 being the highest. An intuitive mapping of these business values to the importance of the data is guided by [7] and given in Table 1.

Business Value	Importance of Data
9	Mission Critical
8	Business Critical
7	Essential
6	Consequential
5	Non-Critical
3-4	Inconsequential
1-2	Disposable

Table 1: Mapping of business value to the importance of the data.

Storage Classification Engine: This component is responsible for mining the capabilities of the available storage subsystems, and classifying them into different tiers of storage quality based on their ability to provide support for objectives such as disaster recovery, performance, availability, etc. The output of this engine is a set of storage classes that provide similar storage characteristics for the data that is stored on them.

Data Placement Engine: This component glues the two classification engines together. It uses the data classes and the storage classes to determine a storage plan for data placement. It recommends which data needs to be placed on which storage resource in order to meet the business needs of the application that uses the data. Thus, it suggests a storage plan that allows for informed use of the storage resources available in the system, and yet allows the data to be accessed efficiently as required by applications.

We note that the classification and placement is done based on some *global objective function* such as disaster recovery, performance, compliance, security, etc., that the administrator can choose. The set of metadata attributes chosen for monitoring, and the policies used for classification change based on the global objective chosen. For example, a much-used temporary log file may be important from a performance perspective but may not be so important from a disaster recovery perspective. ACE allows the administrator to specify an objective function in the beginning to guide the classification and data placement process.

In addition, ACE architecture is modular. Existing classification engines can easily be used in conjunction with other ACE components. The output of other classification engines can serve as an input to the data placement engine to get a data placement plan for a given system.

2.1 Input

The three sources of input to ACE as shown in Figure 1 are as follows:

Knowledge Repository: This repository is a collection of *policies* that encapsulate domain knowledge for data (or storage) classification. Each policy consists of a set of observable attributes of the data (or storage), the corresponding attribute values, and a business value for the data (or storage) that matches these attribute values. Each data/storage object is compared with all the policies to determine which one suits it best. Details of the different policy types are given in Section 2.2.

Observable Attributes: This input comes from mining the attributes of the data files and the capabilities of the storage resources themselves. For example, we can monitor different attributes of the data such as file type, size, last update time, etc. These attributes are used in conjunction with the policies in the knowledge repository to classify the data into buckets of different business values, and storage into different tiers of quality.

User Input: The administrator can provide additional input and hints about how to classify the data. For example, she can give sample files with business values. She can also create new customized policies that can be added to the knowledge repository. Finally, she can override the suggestions from ACE, and modify the data and storage classes or the data placement solution.

2.2 Classification Mechanics

Classification in ACE is done using a simple grouping of all data objects with the same business value into a single class. Business values are assigned to data objects using policies. For the rest of the paper, we refer to this process of business valuation using policies followed by classification as *classification using policies*. In ACE, there are three ways to classify the data and the storage. In this section, we discuss the mechanics of the classification itself and give specific examples of different policies in Sections 2.3 and 2.4. The classification engines mine the attributes of the data or the storage subsystem, and then check to see which of the policies can be used to classify the objects. The policies are defined in three ways depending on the expertise of the administrator as explained below. We explain the different policies for data objects but similar techniques can be used to define policies for the storage resources.

Knowledge-based policies: Knowledge-based policies come pre-packaged with the ACE framework. This information is collected over a period of time in consultation with experts, and is based on experience. ACE will have different set of policies based on the domain of the data that is being classified. For example, data for a medical application like X-rays files on a server will be classified differently (using different knowledge policies) than files that are used by a client for personal purposes. The administrator can also create new knowledge policies on the fly before the classification is done. It is possible that a particular data object does not satisfy all the attribute values given in any knowledge-based policy so it cannot be classified using any policy. In that case, ACE does of the following:

- Uses the policy with the largest fraction of matching attributes. Let the matching policies be $P_1, P_2, P_3, \ldots, P_n$ with number of attributes in each policy as $T(P_i)$. Let the number of matching attributes for the data object with each policy be $M(P_i)$. Then the object is classified using the policy P_i, argmax $\left(\frac{M(P_i)}{T(P_i)}\right)$, provided the number of matching attributes is greater than 50%. If two policies have the same maximum matching ratio, the policy that assigns the higher business value is chosen.
- Assigns some default business value to the data object if none of the policies have maximum matching ratio greater than 50%.

Figure 2 show a screen shot of the list of knowledge policies that are available in the inbuilt knowledge repository of ACE from our implementation. The administrator can choose a subset of these policies for data and storage classification. She can also create a new knowledge-based policy to add to the repository.

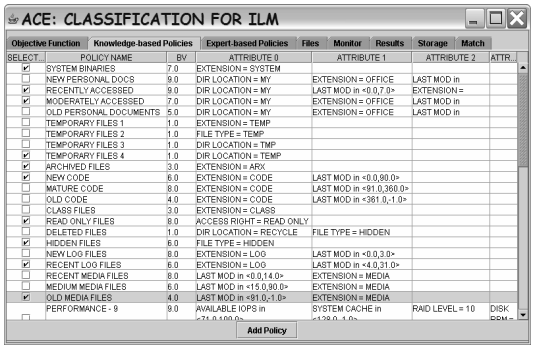


Figure 2: Screen shot showing the knowledge-based policies for data and storage classification.

Expert-based policies: Expert-based policies allow the administrator to rank attributes relatively to formulate a policy. The administrator chooses a set of attributes and assigns relative ranks to them. She also gives scores to different attributes values[1]. These attribute values are then combined and normalized to form a policy function. The metadata attributes are given as input into this policy function, and the normalized output value gives the business value of the data.

For example, suppose the administrator chooses two attributes: A_1 = owner and A_2 = access time. She ranks them as owner being most important and access time being moderately important. ACE internally maps these relative ranks to actual ranks R_1 and R_2. Further suppose that she provides three values for the owner attributes a_{11}, a_{12}, a_{13} which are also given importance in some order. Again ACE will generate internal scores for these values as s_{11}, s_{12}, and s_{13}. Similarly suppose that two values for the access time a_{21}, a_{22} are scored as s_{21}, and s_{22}. ACE generates an internal policy function for a data object d as follows:

BV(d) = $R_1 \cdot ((s_{11} \cdot v(a_{11}) + s_{12} \cdot v(a_{12}) + s_{13} \cdot v(a_{13})) + R_2 \cdot ((s_{21} \cdot v(a_{21}) + s_{22} \cdot v(a_{22}))$

where BV(d) is the business value of data object d. $v(a_{ij}) = 1$ if $A_i = a_{ij}$, and 0 otherwise. Generalizing for any policy, we have:

BV(d) = $\sum_{i=1}^{n} R_i \cdot \sum_{j=1}^{m_i} s_{ij} \cdot v(a_{ij})$

where n is the total number of attributes in the expert policy, and m_i is the number of values for attribute i.

Expert-based policies are flexible but they require expert knowledge to be formulated on the fly. Figure 3 shows a sample expert policy created by an administrator in ACE. The attributes chosen are owner, file extension,

[1]These values can also be ranges for certain attributes such as access time

and access time. Each attribute is given a rank and each attribute value is given a score. ACE uses these ranks and scores to generate a new policy for classification.

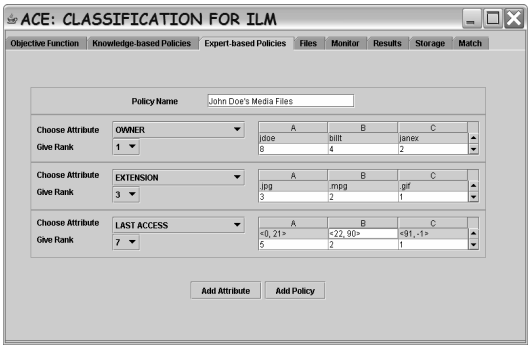


Figure 3: Screen shot showing one expert-based policy for data classification.

Example-based policies: Instead of giving an expert policy, the administrator can give a sample set of files along with their business values. ACE will then mine the attribute values of the sample data set, and the associated business values to come up with a policy that meets the specified business value. The example files serve as a training set to standard machine learning techniques such as regression and decision trees. In this paper we use *linear regression* to learn the policy function from the example file. A regression function is a well-known statistical technique which is used to determine the relationship between one or more independent variables (e.g. metadata attributes) and a dependent variable (e.g. business value) [4].

ACE constructs a regression function using both categorical and numerical attributes with the sample data set, and then uses this function to classify all the data objects. Regression analysis has two primary advantages: one, it always gives an informed business value to the data object unlike knowledge-based policies which may give some default value. Second, it infers information about the metadata attributes that is not obvious to the administrator, and cannot be specified in a knowledge-based policy. For example, the regression analysis may reveal to the administrator the relationships between different attributes (e.g., function may reveal a relation between the attributes of last read time and number of applications using a file), and the relative importance of each attribute for the business value. Thus, the regression function may *learn* some interesting relations between the attributes to give a more informed business value to each data object.

In our implementation, we use example-based policies only for data classification (Section 3.4), and not for storage classification. We plan to develop example-based policies for storage classification as future work.

The policies help to decides the business value of each object based on its attributes. In the first method, each policy is encoded into the system, and in the second and third methods, it is learned on-the-fly using the metadata attribute rankings or the training set provided by the administrator. In the knowledge-based design, the attributes are generic and fixed. Generic attributes are the ones that do no have specific values in given system environment, e.g., create time, access frequency, last modified time. Moreover, the rank of the attributes is decided a-priori. For the expert-based design, the attributes can be generic and/or domain-specific, and their relative ranking can be decided by the administrator. Domain-specific attributes such as owner and application have specific values in a given environment.

2.3 Data Classification Policies

Data classification is done by mining the metadata attributes of the data objects. Here, we consider that the data object is a *file*. Some of the metadata attributes that are used for data classification in ACE are given in Table 2.

ATTRIBUTE	LINUX	WINDOWS
Owner	D	D
Access rights	D	D
Application	I	I
Size	D	D
File type	D	D
Last read time	D	D
Last write time	D	D
Create time	-	D
Extension	D	D
Access frequency	I	I
Number of Applications	I	I
Growth of file	I	I

Table 2: List of attributes that can be mined for data classification. D=Directly available from the operating system. I=Inferred by using internal ACE mechanisms, or a combination of native system APIs.

ACE obtains metadata attributes either by scanning the file system or by parsing trace files of the file system. Most of the attributes that are listed in Table 2 can be determined by a single scan of the system. However, some of the attributes such as access frequency and growth require monitoring of the system over a period of time as explained in Section 2.8.

The policies for data classification depend on both the global objective function chosen as well as the domain of data objects. Policies that classify personal documents and code files for performance are different from policies used to classify medical records for disaster recovery. In Table 3, we give examples of the policies we use in our

evaluation; business values range from $[1-9]$ as given in Table 1, but these values can be parametrized.

So far, we have talked about mining metadata attributes of objects to classify them. Many other techniques classify and cluster files based on content [9, 18, 22]. Our policy-based approach is extensible to also incorporate content-based attributes in the attribute space. Further, the output of any data classification component can be used in conjunction with our storage classification engine and data placement engine to generate a storage resource utilization plan.

2.4 Storage Classification Policies

ACE does storage classification by mining the capabilities of the storage subsystems. These attributes can be collected using some interfaces to the subsystem itself e.g., CIM queries [24]. However, we can use ACE in conjunction with any other software or component which can provide the necessary information about the storage subsystems. Some of the attributes that are used for storage classification in ACE are given in Table 4.

Random I/O capability	Capacity
Utilized capacity	Max Throughput
Current Throughput	Maximum IOPs
Current observed IOPs	Checksum available
Encryption capability	Access authentication
Supported Protocols	WORM capability
Continuous Copy	Snapshot Copy
Physical Dimensions	Power Consumption
Active/Active capability	Firmware Swapping
Multipathing software	MTTR
RAID levels supported	Storage Block Type
Volume resize capability	MTTF
LUN addresses available	Cost
Caching Buffer Size	Latency
Types of disks supported	Cache

Table 4: List of attributes that can be mined for storage classification.

In the case of storage classification, the unit for classification will vary based on the objective function. If the objective function is disaster recovery, the classification is done at the storage controller level. Depending on the kinds of replication mechanisms that a controller support, we can determine what tier of disaster recovery support it can provide. On the other hand, if the objective function is performance, then we can classify the storage resources at the pool level. We can distinguish between different pools based on the RAID level, the underlying disk RPM, the available throughput, etc. For illustrating ACE classification and data placement we only focus on the objective function of disaster recovery in our evaluation.

Table 5 shows some sample storage classification policies. Storage classes range from $[1-9]$ with 9 being the highest quality class, but again these values can be parametrized.

2.5 Data Placement

Data placement involves matching the data classes to the appropriate storage classes. We want to store the most important data on the best-quality storage, and store the least important data in the lowest quality storage. In general, ideal data placement is difficult to accomplish. Using the classes generated by the ACE data and storage classification engines makes placement easier as it qualifies the type of data and storage in the system. We take a simple approach of matching the highest data class with the highest available storage class that has free storage to store the new data. Similarly, we match the moderately important data with the medium quality storage classes, and the least important data with the lowest quality storage classes. If a particular storage class is not available, ACE recommends the next best available storage class which may be better or worse than the ideal storage class.

While doing this matching, it is important to take into account the free space available to migrate the data from one storage container to another. We cannot assume that all the data will be migrated to the suggested location, and hence we can only estimate the available free space on the storage resources.

2.6 Output

ACE shows the output of all the three components engines. The output of the data classification engine displays the different classes of data based on their business values. Each class display the list of files that have the class' business value. The administrator can choose to see the details of any file, and also change the suggested business value to a different one. The storage classification engine displays the different storage classes based on the quality of the storage. Depending on the global objective function chosen, the output shows classified storage subsystems or classified storage pools.

2.7 Putting it all together

Putting together all the different sections explained above, the final flow of the system (from the administrator's perspective) starting with the data classification and ending with the data placement solution is given in Algorithm 1. The the final flow of the system as seen by ACE is outlined in Algorithm 2.

Figure 4 shows the output of the data placement engine after running ACE on a small subset of sample files

[2]Sample values for .CODE are: vbs, vbp, pas, f, c, cc, cpp, hpp, h, tex, m, java, jar, js, jsp, mof, sql, ddl, xml, py, sh, wsdl, mat, fig, zargo, pl, tcl, tk, xsl.

Policy Name	Business Value	Attribute 1	Attribute 2	Attribute 3
Code Files				
New Files	6	CTIME $\in < 0, 90 >$	EXT=.CODE[2]	-
Mature Files	8	CTIME $\in < 91, 180 >$	EXT=.CODE	-
Personal Documents				
Rarely Accessed Docs	5	ATIME $\in < 45, -1 >$	DIR=DOCUMENTS	EXT=.OFFICE
Frequently Accessed Docs	9	ATIME $\in < 0, 7 >$	DIR=DOCUMENTS	EXT=.OFFICE
Moderately Accessed Media	7	ATIME $\in < 8, 45 >$	EXT=.MEDIA	
Medical Data				
Old Files	3	ATIME $\in < 21, -1 >$	EXT=.MEDICAL	-
New Files	9	ATIME $\in < 0, 7 >$	EXT=.MEDICAL	-

Table 3: Table showing some sample data classification policies for different domains. CTIME = Creation time, ATIME = Last Access Time, EXT = Extension. Some of the values such as .CODE and .OFFICE actually represent an array of values.

Policy Name	Storage Class	Attribute 1	Attribute 2	Attribute 3
Disaster Recovery				
DR Tier 1	9	Continuous Copy=SYNC	% Utilized $\in < 0, 50 >$	Active-active=YES
DR Tier 4	6	Snapshot Copy=YES	Cost $\in < 0, 50 >$	Random I/O=YES
Performance				
Highest Performance	9	Avail. BW $\in < 75, 100 >$	Cache $\in < 128, 256 >$	Disk RPM=15
Medium Performance	7	Avail. BW $\in < 25, 74 >$	Cache $\in < 128, 256 >$	Disk RPM=10

Table 5: Table showing some sample storage classification policies for different objective functions.

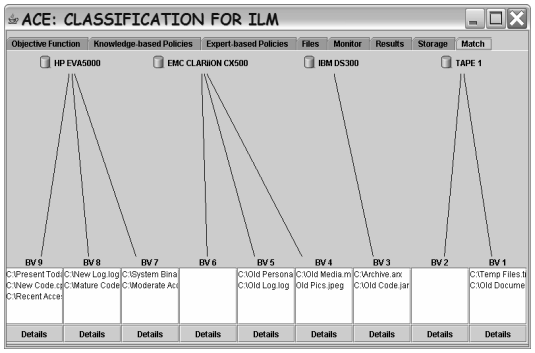


Figure 4: Screen shot showing the output of the data placement engine matching the data classes (for a small sample set of files) with the storage classes.

just for illustration. It displays the different files classified as per their business values and the corresponding storage systems that this data can be stored on. Note that the higher-valued data is stored on high-quality storage as compared to the lower-valued data which is stored of lower-quality storage. For brevity, we only give a screen shot of the output of the data placement engine in this paper.

2.8 Monitoring

One important design issue is how often the system should be monitored by ACE to detect changes in business values of data, and propose new storage solutions. In order to determine the metadata attributes, ACE can either scan the data objects or it can collect (and parse) application-level and file system-level traces. On one hand, we do not want the monitoring to affect the system performance. On the other hand, we want the classification to be as accurate as possible over a period of time. ACE monitors the system either as (i) an ongoing activity periodically, or (ii) by taking *snapshots* of the system at some regular intervals. In the first case, the monitoring frequency is changed on the fly depending on how often the business value of the data is changing. Frequently-changing data is monitored more often than less-frequently changing data. We let the administrator specify how often the monitoring should happen.

Algorithm 1 ACE system flow from administrator perspective

if *create new example-based policy* **then**
  repeat
    pick sample file F
    give business value to F
  until *no more sample files*
else
  choose an objective function O
  choose knowledge-based policies from the ACE set of policies for O
  if *add new knowledge-based policy* **then**
    repeat
      pick attribute A
      pick matching value for A
    until *no more attributes required*
    pick business value for policy
    give the policy a name
  if *create new expert-based policy* **then**
    repeat
      pick attribute A
      give rank to A
      repeat
        pick attribute value (or range) a_i
        give score to a_i
      until *no more attributes values required*
    until *no more attributes required*
    pick business value for policy
choose files to classify
choose monitoring schedule and number of snapshots
repeat choosing knowledge-based and expert-based policies for storage classification
choose storage subsystems for data placement
review and correct data placement solution

Algorithm 2 ACE system flow from ACE perspective

if *create new example-based policy* **then**
  get list of sample files; scan sample files
  if *business values are not given by administrator* **then**
    get objective function O
    classify sample files using knowledge-based policies for O
  generate regression policy function;
else
  get objective function O
  show knowledge-based policies for O
  get selected knowledge-based-policies
  if *create new knowledge-based policy* **then**
    show attributes and attribute values
    get chosen attributes and values
    get business value for policy
    get the policy name
  if *create new expert-based policy* **then**
    show attributes and attribute values
    get attribute ranks
    get attribute value scores
    generate policy function
repeat choosing knowledge-based and expert-based policies for storage classification
scan file system or file system traces for metadata attributes
get list of storage subsystems to classify
classify data files
classify storage resources
determine and display data placement output

Storing snapshots allows administrators to visualize the trends with respect to how the data is being accessed or updated. Users can manually view the trend graphs and use them to validate whether the policies defined for assigning values to the data items are reasonable.

2.9 Performance Optimizations

With the architecture described above, ACE scans each data object for its metadata attributes and then compares it to every policy to determine the business value of the data object. With m policies and n data objects, the worst-case running time of the system is $O(m \cdot n)$. In this section, we describe four potential optimizations to improve the performance of the system. We implement the first two optimizations, and we evaluate their performance in Sections 3.4 and 3.5.

2.9.1 Optimization in the policy domain

One way to improve the performance is to reduce the size of the policy domain by using a *regression function* to represent the policies used for classification. We use the given policy table to classify a given set sample files. We compute the business values of these sample files using the knowledge-based policies. Alternatively, the administrator can even specify the business values of the sample files. We then use the metadata attributes of the files and the computed/given business values to generate a regression function. This single function can now be used to classify the input data rather than compare each data object to every policy. Thus, we reduce the worst-case performance time to $O(n)$. However, using regression may result in some inaccuracies in classification from the original policies as it will only approximate the original knowledge-based policies. Thus, we have to consider the trade-off between the improvement in performance versus the loss of accuracy in classification (Section 3.4).

2.9.2 Optimization in the data domain

Another way to improve the performance is to reduce the size of the data object domain. We propose three techniques to achieve this goal.

File Sampling: Instead of scanning the entire data set, we can sample a small set of files in every directory. We choose a fixed percentage of the total files in every directory, and only scan those files for their metadata attributes. Based on the values of the metadata attributes of the sample set in each directory, ACE uses one of the following methods to classify the rest of the files:

- Pick the policy that is applicable to most of the files in the sample set, and apply the business value from that policy to all the files in the directory.
- Out of all the matching policies, pick the one that has the highest business value and apply that value to all the other files.
- Take an average of the business values all the matching policies and apply that value to all the files.

However, this method involves a trade-off between improvement in performance versus the loss of accuracy in classification. The results of using this approach are sensitive to the original location of the data objects in the object namespace (Section 3.5).

Preprocessing policies: ACE can pre-process the policies to see which attributes of the data objects need to be scanned. Some of the attributes need not be considered if none of the policies are defined using those attributes. This approach helps to improve the scan performance.

Trapping System Events: In a very large system, scanning the entire system to gather all the metadata attributes may be time- and resource-consuming. In such a scenario, the system events such as file creation, deletion, access, etc., may be trapped to constantly monitor any changes to the data set. Such event trapping may degrade the system performance but it may be worth it if the classification performance is of importance. It is useful if the system needs to be up-to-date with the classification information for better resource utilization.

3 Case Studies

To evaluate the ACE framework, we implement and test it using three real data sets from different domains, and several storage resources for classification and data placement. We evaluate the benefit of using ACE, and also demonstrate the improvement in performance using the different optimizations outlined in Section 2.9. We show that ACE is beneficial in identifying important data, and providing large savings in the cost to store this data by suggesting an appropriate data placement.

3.1 Experimental Setup

We run our evaluation tests on an Intel P4 3GHz, 2GB RAM machine running Microsoft Windows 2000 Advanced Server. We implement the ACE framework in about 6200 lines of JAVA code. For mining the metadata attributes of a Windows system, we use Windows API through Java Native Interface (JNI). We use *handle* a freeware tool [13] to determine the application-data relationship. We use a relational database for storing the existing or administrator-supplied knowledge-based policies. We implement ACE such that it scans the target directories, and also accept traces containing the metadata attributes of data objects. Currently ACE supports traces output by NFS tracing utilities (e.g.,`nfsdump`), and unix `ls` and `stat` commands.

We use three real data sets in our evaluation that are shown in Table 6.

NAME	TOTAL FILES	TOTAL SIZE
CODE	12796	316MB
USER	32800	18.4GB
MEDICAL	1.28 million	1461GB

Table 6: List of data sets that we use for our evaluation.

The CODE data set is a local laboratory CVS repository used for the last three years by about 30 users to store their project development code. The USER data set is a collection of personal documents of a single user over a period of five years. This data set includes personal documents, media files, code files, web documents, etc. The MEDICAL data set is a collection of metadata information about anonymous medical files. This data set consists of 1.28 million files which are rarely modified or deleted. This data set is an excellent sample to evaluate ACE because the medical data archiving industry represents the bottom-end of the storage market and is very cost conscious. ACE is an ideal framework to use for this market segment as it can handle a large number of files, and provides storage solutions for cost-effective use of the available resources.

We gather metadata attributes of data objects of the USER data set using ACE's internal scan mechanism, the CODE data set using the attributes gathered by the unix `ls` utilities, and the MEDICAL data set from an anonymous metadata attribute trace supplied by a medical industry company. We use knowledge-based policies to classify each of the data sets. Each data set belongs to a different domain and has different data classification policies; please see some of the sample policies given in Section 2.3. All the data sets and the framework software reside on the same machine.

For storage resources, we use a catalog that contains data about different storage systems and their capabilities. We use different storage classification policies as given in Section 2.4 to classify the storage resources as enterprise, mid-range, low-end, and tape.

We present the following results to evaluate ACE:

1. We show that ACE provides huge cost savings by recommending a data placement solution to store different data on different quality of storage, over the lifetime of the data (Section 3.2).

2. We show that ACE is capable of determining the temporal nature of the business value of data over its lifetime, and change the suggested data placement plan to provide further cost savings (Section 3.3).

3. We evaluate the accuracy of the classification while reducing the size of the policy domain by using regression for example-based policies (Section 3.4).

4. We also evaluate the accuracy of the classification while reducing the size of the data domain by sampling only a few files (Section 3.5).

3.2 Cost savings

In this section, we show the cost benefit of using ACE for data placement versus a naïve data placement of all data on enterprise storage. We use knowledge-based policies to classify the data and storage resources. We use a simple data placement mechanism that maps data files with business values 9 and 8 to enterprise storage, 7 and 6 to medium-end disk, 5 and 4 to low-end disk, and finally $1-3$ on tape.

Figure 5(a) shows the percentage of the size of each data set that can be stored on different quality storage as per the business values of the data. We see that nearly 70% of the CODE data set, and nearly 65% of the USER data set is old, and can be stored on low-end disk. Nearly 70% of the MEDICAL data set can be stored on tape. This graph shows that ACE can identify high business-valued data and recommend an appropriate data placement solution for it.

Figure 5(b) shows the percentage of cost required for different kinds of storage by following ACE-recommended classification and placement as compared to storing all data on enterprise storage. We use standard, industry average numbers for calculating the cost; enterprise storage is $20/GB, medium-end storage is $13.46/GB, low-end storage is $7/GB, and tape is $1.20/GB [6]. We see that there is a significant cost benefit ranging from 50% in the CODE data set to 65% in the MEDICAL data set. The savings in the MEDICAL data set are particularly notable because of the size of the data set; using enterprise storage costs about $\$30,000$ compared to the cost of $\$6,600$ by using the ACE data placement solution.

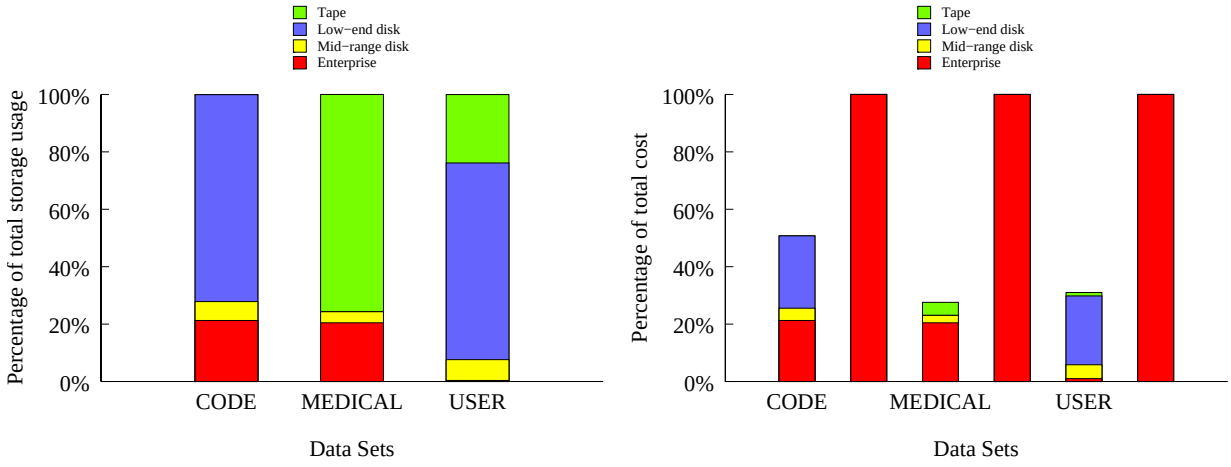
Table 7 gives some details of the output of the data classification engine for the USER data set.

3.3 Temporal Analysis

Next, we evaluate the benefit of using ACE over a period of time as the business value of the data changes. We use ACE to generate the data placement plan at different points of time in the past and the future. Figure 6(a) shows the results of these plans over a period of 18 months for the CODE data set with plans generated every three months. Figure 6(b) shows similar results for the MEDICAL data set over seven weeks with plans generated every week. We see how the value of the data changes over time, as more and more data gets classified as old data. We see that the data grows over a period of time (until the current time) and then it stays constant (as we do not generate new files for plans generated in the future). Even though the data size remains constant, the

Policy Name	Total Files	% Files in Class	Total % of Files	Total Storage (MB)	% Storage in Class	% of Total Storage
TEMPORARY FILES 1	48	72.73%	0.15%	1.90735E-05	0.00%	0.00%
OLD BACKUP FILES 2	9	13.64%	0.03%	3734.101563	99.91%	19.81%
DELETED FILES	1	1.52%	0.00%	0.05403614	0.00%	0.00%
TOTAL BUSINESS VALUE 1	66	100.00%	0.20%	3737.323895	100.00%	19.83%
OLD WEB DOCUMENTS	2736	100.00%	8.34%	14.11757755	100.00%	0.07%
TOTAL BUSINESS VALUE 2	2736	100.00%	8.34%	14.11757755	100.00%	0.07%
CLASS FILES	3776	99.45%	11.51%	8.754601479	1.17%	0.05%
ARCHIVED FILES	21	0.55%	0.06%	737.6234837	98.83%	3.91%
TOTAL BUSINESS VALUE 3	3797	100.00%	11.58%	746.3780851	100.00%	3.96%
OLD MEDIA FILES	20667	92.36%	63.01%	8486.905873	75.31%	45.03%
OLD CODE	1513	6.76%	4.61%	31.24714756	0.28%	0.17%
OLD LOG FILES	4	0.02%	0.01%	0.021150589	0.00%	0.00%
TOTAL BUSINESS VALUE 4	22377	100.00%	68.22%	11269.15622	100.00%	59.79%
OLD PERSONAL DOCUMENTSS	686	71.24%	2.09%	881.5213194	53.63%	4.68%
DEFAULT BUSINESS VALUE	277	28.76%	0.84%	762.3001347	46.37%	4.04%
TOTAL BUSINESS VALUE 5	963	100.00%	2.94%	1643.821454	100.00%	8.72%
MEDIUM MEDIA FILES	522	23.31%	1.59%	322.327776	42.32%	1.71%
HIDDEN FILES	15	0.67%	0.05%	0.001497269	0.00%	0.00%
TOTAL BUSINESS VALUE 6	2239	100.00%	6.83%	761.665102	100.00%	4.04%
SYSTEM BINARIES	78	57.78%	0.24%	552.9692068	92.14%	2.93%
SYSTEM FILES 3	12	8.89%	0.04%	0.847621918	0.14%	0.00%
TOTAL BUSINESS VALUE 7	135	100.00%	0.41%	600.1682491	100.00%	3.18%
MATURE CODE	478	98.35%	1.46%	15.86468124	21.32%	0.08%
RECENT MEDIA FILES	7	1.44%	0.02%	58.54503918	78.68%	0.31%
NEW LOG FILES	1	0.21%	0.00%	0.000547409	0.00%	0.00%
TOTAL BUSINESS VALUE 8	486	100.00%	1.48%	74.41026783	100.00%	0.39%
RECENTLY ACCESSED PERSONAL DOCS	1	100.00%	0.00%	0.06837368	100.00%	0.00%
TOTAL BUSINESS VALUE 9	1	100.00%	0.00%	0.06837368	100.00%	0.00%

Table 7: This table shows the detailed classification of the USER data set using the different data classification policies in this domain. It shows the number and size of some of the files in the business value classes, and some of the policies that are used in each class.



(a) This graph shows the percentage of size of the data sets that can be stored on different quality storage subsystems based on the business value of each file.

(b) This graph shows the percentage of the cost of storage for different types of storage with (left bar) and without (right bar) ACE data placemcent recommendations.

Figure 5: Usage of different storage and effective cost savings using ACE.

storage cost decreases over time as the data becomes less important and is moved to lower-quality cheaper storage. Figure 6 shows how ACE is able to capture the temporal nature of the business value and save storage cost over a period of time.

3.4 Optimizing Policy Domain

In this section, we outline the regression analysis of the USER data set to illustrate the use of example-based policies, and also the reduction in the policy domain for data classification. The goal is to learn a regression function that predicts the business value of a file as a function of file metadata attributes, and then use it to classify other files instead of evaluating the full list of available knowledge-based policies.

We use general linear models for learning the regression function since it allows for the use of both categorical as well as numeric variables [16]. The independent variables of the function are owner, file type, last access time, last write time, create time, file size, and file extension. We test all the independent variables for statistical significance (p-value $< .05$). We also look at 95% confidence intervals for the predictions and make sure that they are 'narrow'. For brevity, we are only presenting the key details here. We use our knowledge-based policies (policies gathered from domain experts) to generate a training and a test set from the USER data set. In this data set which consists of 32800 files, we held 2500 files for validation, and use the remaining for estimating the model using different training set sizes.

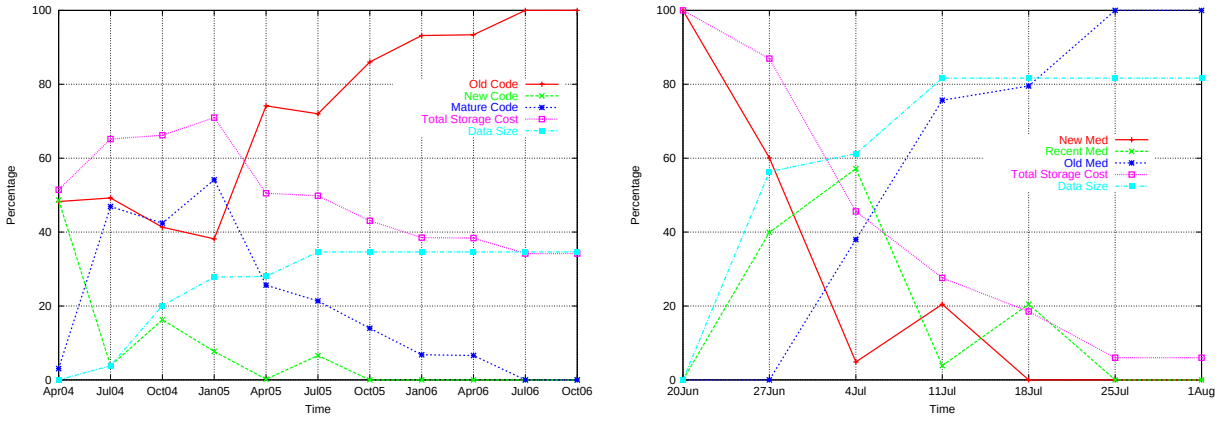
We test the accuracy of our regression function for different training set sizes. Our accuracy metric is the mean absolute percentage error in prediction of the business value for both the training set (estimation period) and the test set (validation period). Figure 7 shows the prediction error of the function for different training set sizes. We see that we get excellent results in the USER data set with the misclassification error less than 15%. The CODE data set does not give as good results for smaller training sets, but it does well with larger training sets. Note that the error does not always go down with increasing training set, since any regression function is always skewed to the data set on which it was trained.

3.5 Optimizing Data Domain

We now explore the performance optimizations for data reduction explained in Section 2.9. We sample files in each directory rather than scan all the files in the system as given in Section 2.9.2. Figure 8 shows the results of using sampling for data classification on the USER data set. We measure the accuracy of the classification in two different ways:

1. Accuracy $= \left(1 - \frac{\text{Number of misclassified files}}{\text{Total number of files}}\right)$. This measure of accuracy simply gives the percentage of misclassified files.

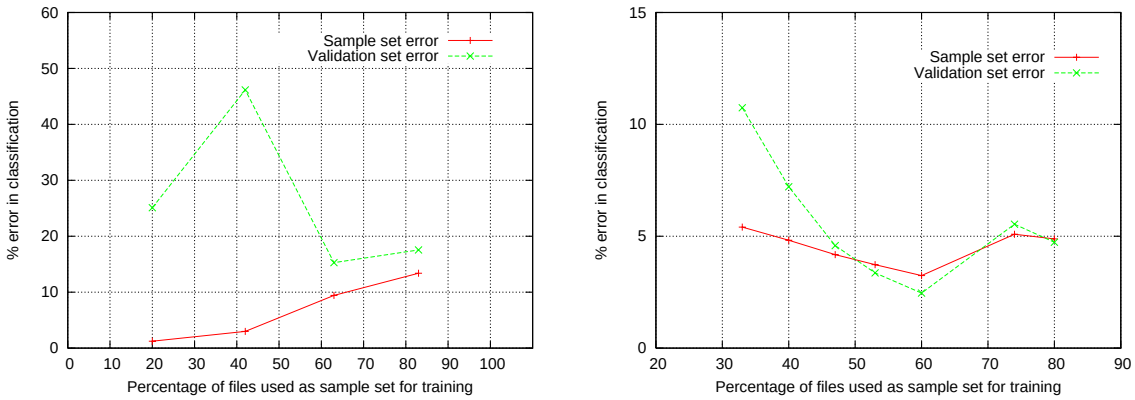
2. Let $I(F_i)$ be the ideal business value and $C(F_i)$ be the computed business value for each file F_i. Accuracy $= \left(1 - \frac{\sum_i |I(F_i) - C(F_i)|}{\text{Total number of files}}\right)$. This measure also indicates the magnitude of misclassification as it measures the distance of the misclassified data from the ideal classification.



(a) Cost benefit for CODE data set.

(b) Cost benefit for MEDICAL data set.

Figure 6: This graph shows how ACE captures the temporal nature of the business value of data over time. The percentage of data with lower business value grows over time and more data gets older. Older data is recommended to be stored on low-end or tape storage resulting in significant cost savings.



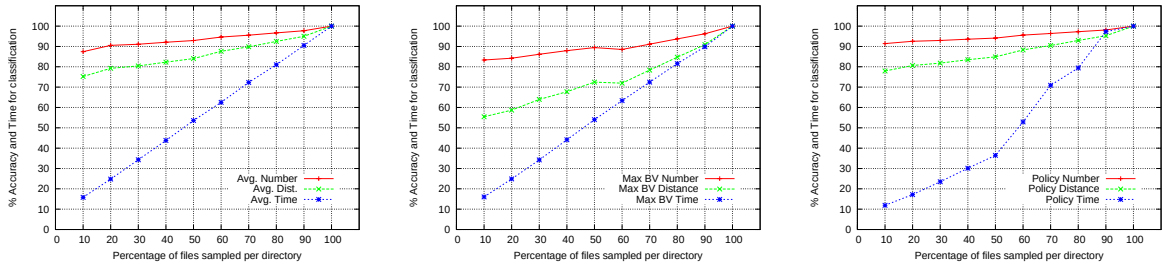
(a) CODE data set

(b) USER data set

Figure 7: This graph shows the classification done using example-based policies and regression function analysis for the two data sets CODE and USER.

Figure 8(a) shows the percentage of time taken and the percentage accuracy of classification using both measures of accuracy when the average business value of sampled files is applied to all the remaining files. Similarly, figures 8(b) and 8(c) show the graphs for maximum business value, and the business value applicable to most files in the sample set. The time taken for the three methods for this data set with 100% sampling is 379 seconds, 374 seconds, and 554 seconds respectively for 32800 files of total size 18.4GB. We see that we get upto 90% classification accuracy with just 10% sampling of files for all but one valuation technique for the USER data set. Further, we reduce the total time to 15% of the time without this optimization while maintaining 85% accuracy. The result shows that if the files are well organized into directories, then sampling of files may be very useful to improve the overall performance of ACE.

We note that we only perform this evaluation on this data set because it is the only one where we scan the system for mining the metadata attributes. Since we use trace files for the CODE and MEDICAL data sets, there is very little time difference between using the data from a part of the trace versus using the whole trace because we need to parse the entire trace in any case. Thus, the trade-off between time and accuracy is not very significant in the



(a) Average business value (b) Maximum business value (c) Highest matching policy.

Figure 8: This graph shows the trade-off of classification accuracy and time performance by sampling only a few files rather than scanning the USER data set. The inaccuracy of classification is calculated in two different ways: (i) by counting only the number of misclassified files, and (ii) counting the distance by which each file is misclassified.

case of traces. We also note that the performance benefit may not be obvious for a small data set like the USER data set, but it can be huge when the data set is very large e.g., petabytes of data.

4 Related Work

ACE is a component in a comprehensive architecture for ILM and draws from several areas such as information valuation, data and storage classification, machine learning, and storage planning. We briefly describe how our work is related to some of these areas.

Data Classification Several approaches exist to classify data based on its content as well as its metadata. Our approach is to solely look at the file's metadata, and hence is different from most of the existing data mining techniques. Some recent examples that look at file metadata for classification include the work done by Ellard et al. [20], and Zadok et al. [28]. The main difference is that ACE classes cannot directly be inferred from the metadata. ACE classes are determined by the business value of data, and hence the key challenge in our case is to quantify the business value of files before they can be classified. The business value of the files can be determined using the metadata attributes and the policies for valuation.

Information Valuation Quantifying the business value of data is hard. Several existing approaches use market-based principles for calculating the cost, revenue, and utility of data, e.g., [12, 21]. However, these approaches are often labor and time-intensive, and preclude the system administrator from making quick storage management decisions for ILM. Ying Chen's work [5] looks at a file's recency and its frequency of use for deriving its value, demonstrating the viability of metadata based valuation. This approach is limited in the set of metadata attributes it uses for information valuation. Erez Zadok et al. [28] use a file's extension, directory and its elastic bit to decide its importance category. This work is perhaps closest in nature to the ACE architecture, and ACE can use some of their data valuation techniques to determine the business value of the data. However, the main difference between the two systems is that ACE not only looks at data valuation but it also provides storage classification, and determines a data placement plan for better storage resource utilization. Further, ACE's valuation uses a richer set of metadata attributes, is applicable to a larger variety of files, incorporates both administrator and application knowledge, and is driven by a high level objective function such as disaster recovery, or performance.

Storage Planning for ILM Hierarchical Storage Management systems, e.g., [23, 27] address the lifecycle management of data by moving the less-frequently accessed files to a slower disk or tape. ACE complements HSM systems by quantifying the importance of files based on a large collection of file metadata attributes, and allowing the migration policies to incorporate both administrator and application knowledge.

ACE is also complementary to the existing work on automated storage planning for disaster recovery, e.g., [17], performance, e.g., [1, 2] etc. ACE data and storage classes can serve as an input to the existing planning tools, allowing them to create more efficient plans. Overall, we believe that ACE is an end-to-end solution that combines both data and storage classification to determine a data placement plan for informed use of the available storage resources.

5 Conclusions and Future Work

In this paper, we have introduced ACE, a new architecture to perform data and storage classification, and data placement using metadata attributes of files and capabilities of storage resources. We have demonstrated that ACE

is beneficial in identifying the business valuation of data and assigning data to the appropriate storage hardware for informed resource utilization. ACE performs well with large data sets, and provides significant cost benefits by determining the data placement solution. We believe that ACE is a first step in a comprehensive solution for Information Lifecycle Management.

With the current ACE design, there are still several interesting open problems. We use a simple approach for data placement which involves matching the highest data classes to the best storage classes, and moving down in the classification hierarchy. However, this approach may not give the optimal data placement. It would be very interesting to see how to optimize the data placement. Currently, we only take into account which application is using what data but ignore the data access pattern for that application. The workload characteristics of the application may give us some more hints about data placement.

Further, in this paper we focus on storage controllers for the storage classification and data placement. In a real system, the storage controller characteristics are not sufficient; what we are really interested is in the *path management* of data. The data has to be stored on the storage that is optimal from an access path point of view starting from the host that accesses the data, through the switches, links, etc., until it finally reaches the storage subsystem. Future work involves optimal path management for the data.

References

[1] E. Anderson, M. Hobbs, K. Keeton, S. Spence, M. Uysal, and A. Veitch. Hippodrome: Running Circles Around Storage Administration. In *Proceedings of Conference on File And Storage Technologies*, Jan. 2002.

[2] E. Anderson, M. Kallahalla, S. Spence, R. Swaminathan, and Q. Wang. Ergastulum: Quickly Finding Near-optimal Storage System Designs. HP Laboratories SSP technical report HPL-SSP-2001-05, June 2002.

[3] Anonymous for Blind Review.

[4] D. E. Bell and A. Schleifer. *Data Analysis, Regression and Forecasting*. South-Western College Pub, Oct. 1994.

[5] Y. Chen. Information valuation for Information Lifecycle Management. In *Proceedings of International Conference on Autonomic Computing*, June 2005.

[6] Cost of storage per byte. In private communication with the market analysts from IBM, 2005.

[7] M. Croy. The Business Value of Data. `http://www.technologyexecutivesclub.com/Articles/artBusinessValueofData%.htm`.

[8] Understanding Data Lifecycle Management. `http://www.veritas.com/van/articles/4435.jsp`, 2003.

[9] I. S. Dhillon and D. S. Modha. Concept Decompositions for Large Sparse Text Data using Clustering. *Machine Learning*, 42(1):143–175, Jan. 2001.

[10] EMC Documentum. `http://www.documentum.com`.

[11] Gartner. Emerging Technology: Keeping Storage Costs Under Control. *Network Magazine*, Oct. 2002.

[12] R. Glazer. Measuring the Value of Information: The Information-Intensive Organization. *IBM Systems Journal*, 32(1):99–110, 1993.

[13] Process Explorer. `http://www.sysinternals.com/Utilities/Handle.html`.

[14] HIPPA. `http://www.hippa.org`.

[15] International Financial Reporting Standards. `http://www.iasplus.com/standard/standard.htm`.

[16] R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley- Interscience, Apr. 1991.

[17] K. Keeton, C. A. Santos, D. Beyer, J. S. Chase, and J. Wilkes. Designing for Disasters. In *Proceedings of Conference on File And Storage Technologies*, pages 59–72, Apr. 2004.

[18] D. Koller and M. Sahami. Hierarchically Classifying Documents Using Very Few Words. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 170–178, July 1997.

[19] J. T. McArthur. Storage Networking: Business Drivers for SANs. *IDC*, 2003.

[20] M. Mesnier, E. Thereska, G. R. Ganger, D. Ellard, and M. I. Seltzer. File Classification in Self-* Storage Systems. In *Proceedings of International Conference on Autonomic Computing*, pages 44–51, May 2004.

[21] D. Moody and P. Walsh. The Value of Information: An Asset Valuation Approach. In *Proceedings of Seventh European Conference on Information Systems (ECIS '99)*, June 1999.

[22] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[23] SGI InfiniteStorage Data Migration Facility (DMF). A White Paper. `http://www.sgi.com/pdfs/3631.pdf`.

[24] Storage Networking Industry Association. `http://www.snia.org/home`.

[25] IBM Tivoli Storage Manager. `http://www-306.ibm.com/software/tivoli/products/storage-mgr/`.

[26] UK Companies Bill. `http://www.dti.gov.uk/companiesbill/`.

[27] VERITAS NetBackup Storage Migrator for UNIX v4.5. A White Paper. `http://eval.veritas.com/mktginfo/products/White_Papers/Data_Protection/%smu_unix45_wp.pdf`.

[28] E. Zadok, J. Osborn, A. Shater, C. P. Wright, K. Muniswamy-Reddy, and J. Nieh. Reducing Storage Management Costs via Informed User-Based Policies. In *Proceedings of the 12th NASA Goddard, 21st IEEE Conference on Mass Storage Systems and Technologies (MSST 2004)*, pages 193–197, Apr. 2004.