RC22803 (W0305-091) May 19, 2003
Computer Science

IBM Research Report

A Peer to Peer System for Data Management

Dinesh Verma
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

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

A peer to peer System for Data Management

Dinesh C. Verma

IBM T J Watson Research Center,
19 Skyline Drive, Hawthorne, NY 10532
dverma@us.ibm.com

Data Management Systems in enterprises constitute a significant portion of the total cost of management of data in enterprise systems. Maintaining data with a high degree of availability and reliability is typically done by having a centralized backup system that to maintain backup copies of data present on the workstations of enterprises. Maintaining a large dedicated backup server for data management requires a highly scalable network and storage infrastructure, leading to a major expense center. With the current trends in workstation disk storage, an alternative peer-to-peer paradigm for data management can provide an approach that provides equivalent performance at a fraction of the cost of the centralized backup system. In this paper, we present the architecture of a peer-to-peer system for data management, and evaluate its effectiveness as compared to a central approach

1. Introduction

One of the key IT challenges in any enterprise environment is to have an efficient and reliable data management infrastructure. A typical enterprise IT environment consists of many computing elements such as workstations, personal computers, laptops, departmental servers, etc. Each of these computers has a large amount of data generated during its normal course of operation. The data on these computers is often critical to the operation of an enterprise, and the loss of data can result in reduced productivity, lost revenues and an interruption in the normal flow of business processes.

The loss of data is not an unusual phenomenon, and occurs due to various causes, such as disk crashes, accidental erasures, user mistakes, viruses, etc. Data management systems provide the ability for users to recover data that may be lost due to any reason. The key functions of data management systems include making automated periodic copies of data in a reliable manner, and restoring data from the backup copies when required.

The traditional data management approach within an enterprise has been to operate a backup server to maintain backup copies of data across the enterprise. Each backup server handles the backup functions of a set of computers (backup clients), and must have the capacity to store all of the data present at all the clients assigned to it, as well as provide a very high degree of availability. Consequently, backup servers tend to be high-end expensive machines with high capacity data storage devices. Software

products that provide automatic backup and restore functions are available from many vendors, some examples being Veritas Backup Exec [1] and Tivoli Storage Manager [2]. The operation and maintenance of backup servers is a large expense for most enterprises. It is estimated that enterprises spend three dollars in order to manage data for every dollar spent on new storage hardware [3]. A data management system that is able to reduce the cost of data management would help in reducing IT costs in enterprises significantly.

The rise of peer-to-peer systems promises an alternative approach towards data management which can offer equivalent functionality at much lower costs. Most of the personal computers, as well as laptops operational in an enterprise have a large disk-space which is often sparsely used. Industrial surveys indicate that up to 60% of storage capacity in an enterprise typically remains unused [3]. With the recent advances in the capacity of the disks on PCs and laptops, the amount of unused storage on the clients is likely to be much higher than the above figure. The excess storage capacity available in the devices can be exploited to provide data backup services to other devices in the enterprise. If all the machines in an enterprise cooperate to provide backup service to each other, the need for maintaining an expensive backup server would be eliminated, and a much lower cost solution for data management would be developed. Needless to say, client workstations are not as reliable as the backup server, and the key challenge in a peer-to-peer approach would be to ensure that the collection of several less reliable machines results in a highly reliable system from which a backup copy is almost always available for restoration.

In this paper, we describe a peer-to-peer system for data backup and recovery services, and show how it can be made highly reliable and available. We analyze the availability and performance of this system, and show that this system can obtain performance levels comparable to that of traditional backup systems.

The rest of the paper is structured as follows: In section 2, we discuss the structure of a traditional backup system. This is followed in section 3 by the design of a peer-to-peer system for backup and restoration. In section 4, we analyze the availability and reliability constraint of the different approaches to building a data management system. Finally, we present our conclusions and areas for future investigation.

2. The traditional data management system.

We assume that a traditional backup system operates in an environment where there are multiple backup clients that are backed up at a single backup site. For scalability reasons, the backup site may consist of a cluster of backup machines rather than a single machine. Since the backup site has to handle storage needs of a large number of clients, it is usually implemented as one or more high-end servers running with a very large amount of storage space, which may be obtained by a mixture of tape-drives, storage area network devices, or using network attached storage. However, the cluster essentially appears as a single server to the backup clients.

Large enterprises may have several backup sites, with each backup site supporting a subset of client machines. The partitioning of client machines among different backup sites is determined primarily by the proximity of a client to the backup site. In

many cases, the backup sites act as peers to each other, and providing mirroring capability to provide disaster recovery support for storage backup functions. An example of such mirroring capabilities is provided by the IBM TESS [4].

Each backup client in the system would typically have a copy of client software which provides the ability for a user to manually backup and restore files from the servers. Additionally, the client software would typically contain a scheme to run automated backups at scheduled regular intervals, e.g. each client may automatically be backed up every six hours. In order to speed up the process of backing data, backup may be done in an incremental mode where only changes to an existing set of file since the last backup are sent to the server. The metadata associated with a backup, e.g. the last time when a file is backed up, is usually maintained on the backup server. The backup client would check if the file had changed since the last backup time, and if so send the modified files to the storage server. In the case of full backup, all the files are copied over to the new server.

The bulk of the storage costs associated with data management in an enterprise arise due to the need to manage and operate the backup site, which needs to be highly scalable, and available round the clock. The costs associated with establishing, maintaining, staffing and operating the backup server make the backup storage many times more expensive than the cost of the client storage.

The high costs associated with backup of servers leads to a paradoxical situation where the total cost of ownership of a megabyte of storage increases as the capacity of client disk increases. As the capacity of disks on individual workstations increases, the enterprise needs to provision for increased storage at the backup site. On one hand, users and application developers feel that one need not worry about a few extra megabytes on their PCs due to the large capacity of the disks. On the other hand, they come under increasing pressure from the enterprise IT staff to reduce the sizes of actual storage usage, mailboxes, etc. in order to reduce the costs at the backup site. Since backup sites tend to be upgraded at a much slower pace than the turn-around time of PCs and laptops, we are headed for an era where storage space at backup servers is going to be expensive and in short supply, while there is a surfeit of unused disk space on each individual user's machine.

3. The Peer to Peer Data Management System

Peer to peer systems have been used successfully for the purpose of sharing and exchanging files on the Internet [5] [6], as well as to provide for large distributed storage systems [7]. Peer to peer systems can be classified into two broad categories, the first category building applications on top of an application level multicasting and query-searching mechanism, and the other category implementing systems for distributed hash-tables implemented over an overlay space. When searching for a file in the multicasting peer-to-peer system, the querying machine sends a query which is broadcast/multicast to all of the peers who search for the presence of a file matching that query on their local system. In a distributed hash table, the file name is mapped

on to a key, and an efficient method to look up that key in a distributed environment is implemented. The multicasting approach provides for a more general type of query, and has a proven record of scalability and robustness in the wide-area network.

If most of the users in an enterprise have machines that have relatively low disk utilization, they could provide backup and restore functions to their peers in a distributed manner. A peer-to-peer paradigm for data backup and restoration would eliminate the need for a backup site, and would result in substantial cost savings for the enterprise. Using the peer-to-peer backup paradigm, each file is copied over to another peer in the system, rather than to a central backup site. However, the peers are not likely to have the same degree of availability as the backup site. Thus, in order to obtain a higher availability for each file, a file would need to be copied to more than one peer in order for it to become available.

The peer to peer data backup architecture on any machine creates an independent area on each peer that is used for data backup from other peers. The user of the peer can specify configuration properties like the maximum fraction of the disk space to be used for backup functions and the location of the file.

3.1 System Architecture

In the peer-to-peer system for data management, there would be a common software installed on each of the computers within an enterprise. The common software would provide the ability to backup and restore files as needed by a workstation client. The structure of the common software that needs to be installed on each of the machines is shown in Figure 1, and consists of the following components:

Basic P2P Broadcast Mechanism: These are the basic components available as the building blocks for performing an application level broadcast on a peer to peer network. The basic P2P Search mechanism provides the ability to search for a file with a given name and set of attributes on a peer to peer network. Libraries providing this capability are generally available as components in most peer to peer software distributions.

The Peer Searcher: The peer searcher is built atop the basic P2P search mechanism, and is used to search for a peer who would be a suitable candidate for maintaining backups of a specific file in the system. When a file needs to be backed up, the peer searcher component floats a query on the peer to peer network, looking for possible peers who should receive a backup copy of the file. The peer searcher components on other machines respond to such queries. The peer searcher would then select a subset of the responding peers as suitable copies for replication and backup.

The File Searcher: The file searcher is built atop the basic P2P search mechanism and is used to search for peers who are holding a backup copy of a file and are currently available.

The backup/restore manager is the component responsible for the completing the backup of the files on the local machine, and restoring a file from its backup copy, when needed.

The Properties Manager: is a component which keeps on tracking the properties of a current machine to assess its suitability to act as a backup copy for a peer. Properties tracked by the properties manager keeps track of aspects such as when the peer is typically up and connected to the enterprise network, the disk utilization on the system, and the speed of the processor on the system.

The data manager is the component which maintains copies of backup files on the local system. This component is largely similar to that in traditional backup systems, but has the additional role of maintaining the files securely so that the backed up data is only visible to owner of the data.

The schedule manager is responsible for initiating the periodic backup of the files on the local machine to remote machines on a regular periodic basis. The function is unchanged from that in corresponding traditional backup clients running on an automated schedule.

Each of these components (except the basic p2p search mechanism) is described in more details in subsequent sections.

3.1.1 The backup/restore manager

The backup/restore manager is responsible for creating the backup of a file system to one or more peers within the system, as well as for restoring the lost copies of a file in the system. In our architecture, each file is copied independently of the other files on the system. This ensures that each file is copied to an independent set of peers.

In order to copy a file, the backup/restore manager first searches for a set of suitable peers for backing up the file by contacting the peer search module. The peer search module selects a set of peers and returns them to the backup module. The backup agent then contacts the peer to create a copy of the file on each of the selected peer. If the other peer already has a previous copy of the file, an incremental backup of changes to the file is made to the peer. Otherwise, the full file is copied to the backup server. The file can also be encrypted as described in the security considerations section later on.

When a copy of a file is to be restored, the date of last restoration and name of the file are sent to the file searcher module. The file searcher module looks for a copy of the file that is most recently available copy among all the peers. That copy is then used to create the restored version of the copy.

The backup/restore manager is also responsible for classifying files on the local system into different categories. File classified as application files or temporary files will not be backed up at all. Other files would be backed up in the usual fashion. Such classification is a normal feature of most traditional backup systems.

3.1.2 The Peer Searcher

The peer searcher is responsible for locating a few peers for each file which can act as potentially good sites to create a backup copy of the system. The peer searcher is easier to implement on a peer-to-peer search mechanism that implements application level multicast than on a distributed hash table paradigm.

In order to search for a suitable peer, the peer searcher module floats a broadcast query using the basic peer-to-peer broadcast mechanism. The broadcast query contains the name of the file being backed up, the size of the file, the name of the local machine, and the uptime cycle to which the current machine belongs. The uptime cycle of the machine is described in the Properties Manager Section of this paper. The peer searcher modules on the other peer receives the query, and compose a response consisting of the following properties: the uptime cycle of the peer, the amount of free space on the peer, and a Boolean flag indicating if a copy of the file already belongs in the peer being contacted. The peer searcher module on the originating node collects all the responses, and assigns a weight to each of the responding peers. The weight is computed so that peers with an existing copy of the file are preferentially selected, peers with same uptime cycles are preferentially selected, and peers with smaller disk spaces are preferentially selected. The set of peers is then used to create backup copies of the file.

3.1.3 The file Searcher

The file searcher module is responsible for finding the existing copy of a file to be restored. The name of the local peer and the file name are used as the keys to locate the file on the existing peer to peer network. Each peer that has a copy of the file being searched responds to the original peer, including the time when its backup copy was created in the response. The querying peer selects the backup peer whose copy is the latest one prior to the restoration time. The information is passed to the backup/restore manager which actually restores the file. The file searcher can be implemented over the distributed hash table paradigm (using the identity of the node and file-name as keys to the hash table), or over a broadcast/multicast paradigm for building peer-to-peer systems.

3.1.4 The properties Manager

The properties manager module is responsible for keeping track of the characteristics of the local machine on which it is running. The properties manager keeps track of the times during the day when the local machine tends to be up. All machines maintain a uptime cycle property. The uptime cycle is a vector of 24 numbers, each being a numeric probability that the machine will be up during that time of the day. The probabilities are computed by the properties manager keeping track of whether the machine was up or down during the specified hour over the duration of the previous month. If the statistics are not available for a month, the statistics is computed over available data if more than 7 days worth of data is available. If sufficient data is not

available, the uptime cycle is initialized to contain a probability of 1 during 9-5 local time, and a probability of 0 during other times.

In addition to the uptime cycle, the properties manager also keeps track of the amount of disk which is allocated on the local peer for the task of backing up files from other peers. As the disk allocated approaches saturation, the peer is less likely to respond to requests from other peers to backup copies of files from them. The properties manager also maintains an indicator of the type of network connectivity that the peer is likely to have to other servers. This is determined by looking at the characteristics of the network interface that the server has active at each hour. Hours when the peer has slow network connectivity (e.g., the only interface active is a dial-up modem), the uptime cycle is marked to indicate the machine has a low probability of being available.

3.1.5 The Data Manager

The data manager is responsible for keeping copies of the backup files on the local disk. The data manager maintains the timestamp when each file is backed up. The timestamp of the backup copy is computed according to the clock of the peer which contained the original copy. The data manager also maintains an index of all the files that are backed up on the local peer, along with its size.

Files that are backed up using incremental differences from the older versions may be stored locally with a listing of the incremental changes, rather than the full version of the files. The data manager is responsible for managing the differences and delivering the complete file to any requesting user.

The data manager keeps all files in a compressed format to reduce its storage requirements. When a file to be archived is encountered for the first time, the data manager also checks to see if the file is identical to another file of the same name from another system. If the file is identical, then only the metadata (timestamp, owning node, etc.) information is created for the new file, with only a pointer made to the copy of the existing file.

Most enterprises maintain a limited document retention policy in order to contain the amount of storage needed at the backup site. Thus, files would typically not be backed up beyond the period of a few years. Some types of data, e.g. billing records, accounts records, tax-related information, are maintained for a larger period of time, e.g. 7 years, depending on government regulations and operating practices. The data manager maintains the time-period for which a file needs to be maintained in the metadata, and files that are past their document retention period are eliminated from the backup.

3.1.6 Schedule Manager

The schedule manager is responsible for scheduling automatic backup of the contents of the local file system at regular intervals. The function is similar to that in traditional data management systems, and implemented using existing system scheduling mechanisms.

3.2 Security Issues

One of the issues with creating backup data at different peers is that of security. Some of the files present on a user's workstation are of secure nature, and should not be visible to other users. However, backing up the files to another peer may make some of the sensitive data become visible to another user. This vulnerability is a unique problem in the peer-to-peer approach, since a centralized backup system usually is built so as to provide isolation among different users.

In order to address the security concerns, a peer-to-peer system may encrypt a file being copied over for backup using a key known only to the original user of the machine. In order to encrypt a file, an independent secret key for encryption is generated for each file. The key generation is done by a deterministic algorithm which takes four inputs (i) a password specified by the user (ii) the identity of the system where the file is created (iii) the name of the file being backed up and (iv) the timestamp on the local system when the copy is being made. When a backup copy is made, the peer with the backup copy has the information about all the parameters (except the first one) as the metadata maintained in its data manager. Thus, the originating host, which knows the first parameter, can regenerate the key for the file during restoration period and retrieve the file. The algorithm for key generation can be made so as to generate strong cryptographic keys. A typical key generation algorithm could take the concatenation of the four input parameters, compute the exponent of the concatenated binary number, and then take the result modulo a maximum key size as the secret key for encryption. Any secret key based encryption scheme can then be used to encrypt the file.

The one drawback in the approach is that a user forgetting their password for backup in the system will have no way to recover or reset the lost password. In order to assist them in recovering the password, the system maintains a known text file which is then encrypted and stored locally. These two files are not copied to the other peers. When a user forgets his password, the system provides utility tools which would help him recover the password by trying to compare different possible combinations against the known file. The approach would be time-consuming and will only work if the known file is not corrupted on the disk.

Another issue with user passwords is that the password can not be changed since that would not match with the keys used for backing up the previous files. In order to address that issue, the data management system maintains a local file containing all of the previous passwords and the time-duration they are used. This local file is always encrypted using the current key, and backed up to other machines using the standard replication mechanism. When a user changes the password, the old password is appended to the password file, and the password file re-encrypted using the new key which would be generated using the new password.

3.3 Hybrid Data Management Approach

The approach to handling security and metadata management in peer-to-peer systems is not as elegant as it is in centralized backup systems. Centralized backup systems allow a easy way to manage user passwords allowing users to reset them as needed, without a need to maintain a history of older passwords used. A good way to obtain the advantages of centralized security management and the cost-savings of a peer-to-peer data backup systems would be to used a hybrid approach.

In the hybrid approach, the metadata associated with each file is maintained at a central repository, while the actual files are maintained at peers. The central repository is also responsible for managing the passwords of the users. The central repository uses a public-key cryptographic approach to encrypting the files on the peers. All files are encrypted using the public key of the repository. Thus, the issues of key management are restricted simply to managing the keys of repository.

The repository also maintains an index of the peers who have the backup copy of the different files. Thus, the repository has a database which can be quickly searched to determine the set of peers that may have a copy available. The restoration process can be done much faster since the peers containing the backup are now easily identified and can be located without a broadcast query search on the peer-to-peer overlay.

All backup and restoration requests are made to the repository. The repository authenticates the identity of the user making the request, and searches for a suitable set of peers for the machine. The repository identifies the peers that will make the backup copy, and the files are copied directly between the peers. After the successful backing up of a file, the meta-data information at the repository is updated.

The hybrid approach requires a central repository, although this repository would require much less bandwidth and data storage than a full-fledged centralized backup solution.

4. Evaluation of Peer to Peer Approach

In this section, we examine the effectiveness of the peer-to-peer approach with some simple system models. The system models we present are more akin to back-of-the-envelop calculations rather than a formal system performance evaluation model. However, they would help in identifying the situations where the peer-to-peer data management approach would be better than the centralized approach and where it would be worse.

In our system model, we assume that there are N machines in the system, and that they are randomly connected. We assume that each of the machines has a disk with the capacity of D and that each machine has a fraction f of its capacity used to create local files. We further assume that b copies of each file are made in the case of the peer-to-peer approach. Furthermore, each client has the probability p_c of being available at any given time, and that each client operates independently of the other clients.

We assume that all optimizations that can be made for data management in peer to peer systems can be mimicked in a centralized server approach, and vice-versa. Thus, the key comparison is between the storage space and bandwidth required for the centralized solution as compared to the peer-to-peer approach. Let us assume that the centralized backup solution provides an availability of p_s.

With b copies being made of the system, a client would be able to recover a file when needed as long as one of the b peers with a copy of the file is available for it to recover from. Thus, the peer-to-peer system would have a reliability better than that of the centralized system, as long as

$$p_s < 1 - (1- p_c)^b$$

which is equivalent to :

$$b > \log(1- p_s)/\log(1- p_c)$$

Figure 2 shows the typical number of copies that would need to be made with some typical availability numbers of clients and the backup server. The different bars show the number of copies needed to obtain the availability expected of a centralized backup system for a given reliability of the clients. For a client reliability of only 90%, 4 copies of a file are able to obtain the availability equivalent to that of a backup server with 4 9s of availability. For a more reliable client, which is available 99% of the times, one could obtain availability equivalent to 6 9s at the centralized server with only 3 copies.

The additional copies of servers come at a cost, since the total storage capacity needed in the peer to peer approach is higher than that of the centralized server. Since b copies of each file are being made, the space required is b times that of a centralized backup server. In order for the peer-to-peer approach to be successful, the peers must have adequate disk-space to handle the requisite number of copies. This requires that the following relation hold true:

$$b * f < \alpha$$

where α is a number less than 1 due to practical considerations like having enough space capacity of clients, or the need to make multiple versions of a file. Thus, a peer-to-peer approach would be feasible as long as

$$\log(1- p_s)/\log(1- p_c) < \alpha/f$$

This would indicate the peer-to-peer approach for data management is feasible provided the utilization of the disk storage at each of the peers is reasonably small. For a utilization of each client of about 20%, and an α of 0.6, Figure 2 would imply that one could obtain 3 9s of availability using clients with an availability of 0.9, and one could obtain 6 9s of availability using clients with an availability of 0.99. If the trend of client workstations with large disk capacity and relative little usage continues to hold, it would appear that the availability and reliability of the peer-to-peer approach would be comparable to centralized backup approaches for data management.

5. Conclusion

In this paper, we have presented an peer-to-peer approach to managing data in an enterprise. The approach provides for a way to backup and restore data in enterprise systems without the need for a centralized backup server. We have presented the architecture of a system that would implement this approach, and analyzed the feasibility of the approach using some simple assumptions. The analysis indicates the approach would work well for systems where there are many clients with large amount of space disk capacity. Under the present trend of having PCs and laptops with larger amount of disk storage, this approach appears to be an effective low-cost method for managing data in an enterprise.

References

[1] Veritas Software, http://www.veritas.com/products/listing/ProductListing.jhtml
[2] IBM Tivoli Storage Manager, http://www-3.ibm.com/software/tivoli/products/storage-mgr/ .
[3] Elizabeth Clark, *Emerging Technology, Keeping storage costs under Control*, Network Magazine, October 2002, http://www.networkmagazine.com/article/NMG20020930S0004.
[4] B. Mellish, P. Chudasma, S. Lukas and M. Rosichini, *IBM Enterprise Total Storage Server*, IBM Redbook SG-24-5757, http://www.redbooks.ibm.com/pubs/pdfs/redbooks/sg245757.pdf.
[5] Gnutella home page, http://gnuella.wego.com
[6] Kazaa, http://www.kazaa.com
[7] Sean Rhea, Patrick Eaton, Dennis Geels, Hakim Weatherspoon, Ben Zhao, and John Kubiatowicz, Pond: the OceanStore Prototype**,** *Proceedings of the 2nd USENIX Conference on File and Storage Technologies (FAST '03)*, March 2003
[8] Jabber Conversations System Architecture, http://www.jabber.org

Figures

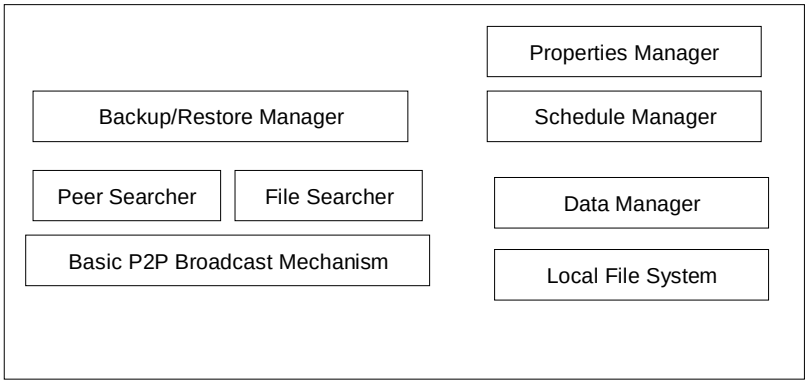

Figure 1. Structure of P2P data Management Software

Figure 2