The person charging this material is responsible for its return to the library from which it was withdrawn on or before the Latest Date stamped below.

Theft, mutilation, and underlining of books are reasons for disciplinary action and may result in dismissal from the University.

UNIVERSITY OF ILLINOIS LIBRARY AT URBANA-CHAMPAIGN

ENGINEERING
CONFERENCE ROOM

JUL 2 1979
JUN 11 RECD
FEB 27 1991
MAR 05 RECD

L161 - O-1096
Center for Advanced Computation
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

CAC Document No. 115

Distributed Data Management in the WWMCCS Environment

by

Peter A. Alsberg

April 23, 1974
Distributed Data Management in the WWMCCS Environment

by

Peter A. Alsberg

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

April 23, 1974
# TABLE OF CONTENTS

1. Introduction .................................................. 1

2. The Self-tuning System ........................................ 2
   2.1 CAC production data management experience ............ 2
   2.2 The IRIS experience ...................................... 2
   2.3 Extension of the IRIS example ............................ 4
   2.4 Increasing data utilization ............................... 6

3. Some Data Management Research Areas for WWMCCS .............. 8
   3.1 Automatic compression .................................. 8
   3.2 Query tuning ........................................... 9
   3.3 Clustering ............................................... 10
   3.4 Restructuring .......................................... 12
   3.5 Back-up and recovery ................................... 13
   3.6 Load Leveling and data base distribution .............. 13
   3.7 Data representation .................................... 15
   3.8 Low error update algorithms ............................ 16
   3.9 Data management and display in an intelligent terminal ... 16

4. WWDMMS Enhancements ......................................... 18
   4.1 Subsystem command ....................................... 18
   4.2 Upward compatibility ................................... 18
   4.3 An implementation plan .................................. 19
      4.3.1 Phase I - subsystem & compress .................... 19
      4.3.2 Phase II - tune & measure .......................... 20
      4.3.3 Phase III - clusters & logging .................... 20
      4.3.4 Phase IV - network operations ..................... 20

5. Conclusions .................................................. 21

6. Bibliography .................................................. 22
**LIST OF FIGURES**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>figure 1</td>
<td>Full Expression Evaluation</td>
<td>3</td>
</tr>
<tr>
<td>figure 2</td>
<td>Pruned Expression Evaluation</td>
<td>3</td>
</tr>
<tr>
<td>figure 3</td>
<td>Dynamic Expression Tuning</td>
<td>5</td>
</tr>
<tr>
<td>figure 4</td>
<td>Cost for Three Evaluation Schemes</td>
<td>5</td>
</tr>
<tr>
<td>figure 5a</td>
<td>Unclustered Data Base</td>
<td>11</td>
</tr>
<tr>
<td>figure 5b</td>
<td>Clustered Data Base</td>
<td>11</td>
</tr>
</tbody>
</table>
ABSTRACT

The Distributed Data Management problems of the World Wide Military Command and Control System (WWMCCS) ADP network are discussed. The application of recently developed data compression and query tuning technology to this problem is described. The concept of a self monitoring and self restructuring data management system is described. The self organizing system, if successful, would have a significant impact on the ADP network performance and reliability. Initial areas of research and development are identified; a research and development program to address those areas is presented; and a plan is described to integrate the research and development program with the ongoing activity to develop a World Wide Data Management System (WWDMS).
Distributed Data Management in the WWMCCS Environment

1. Introduction

This paper is an outgrowth of conversations between JTSA and the Center for Advanced Computation personnel concerning data management techniques which are applicable to the large interactive network data base.

Section 2 discusses the self tuning system. The self tuning system observes its own performance and the pattern of access to its data base. Based upon its own performance and user access patterns, it periodically restructures itself, changes coding schemes, or otherwise modifies its data base or processing algorithms to improve its own efficiency. The experience of the Center for Advanced Computation with self tuning systems is briefly described. Areas for further research are discussed.

Section 3 discusses research and development areas that promise significant impact on WWMCCS performance and survivability. An R & D program aimed at these areas should produce the base technology for network data management systems.

Section 4 discusses how a research and development program based on Section 3 could be used to provide early fall-out in the WWDMS program. A 4 phase scenario is described for the orderly enhancement of WWDMS. The current single-site WWDMS using conventional data management technology would be transformed into a fully distributed, multi-site WWDMS using self-tuning data management technology.

Section 5 summarizes the technical discussion. Section 6 contains a bibliography of the relevant data management literature. This bibliography is the result of a preliminary literature search by Ms. Suzanne Sluizer.
2. The Self-tuning System

2.1 CAC production data management experience

The Center for Advanced Computation has over four years of experience in the design and implementation of user-oriented data management systems. The NARIS and IRIS information systems for planners both handle very large land use and natural resource files interactively. They have successfully made the transition from the research environment to the user-supported production environment. Other examples are the Monica interactive statistical system and an interactive accounting system for handling University departmental accounts. We are currently implementing a distributed data management and statistical analysis system on the ARPA network. One of the design objectives of this distributed system is that it be integrated with several existing statistical systems. The existing systems run at several different sites on the ARPA network under different operating systems and vendor equipment.

2.2 The IRIS experience

The IRIS system had to handle particularly large data bases and provide conversational access to these data bases at the same time. In order to meet these objectives two important techniques were developed which have application to WWMCCS.

The automatic data compression techniques reduced the size of the data base by a factor of 4. Since the critical parameter in the large data base management system is I/O time, the 4:1 size compression produced a corresponding 4:1 reduction in the amount of I/O time it took to physically pass data through the machine. That, in turn, reduced response time by a factor 4.

The dynamic query tuning facility is a bit more subtle. In the IRIS application, it was expected that complicated expressions would be evaluated in order to qualify a record for further analysis or retrieval. Conventional data management systems normally fully evaluate a Boolean expression before qualifying or disqualifying a record (Figure 1). A very few data management systems are clever
Full Expression Evaluation

figure 1

Pruned Expression Evaluation

figure 2
enough to notice that if the left of an "and" operator in the Boolean expression is false, then the result of the "and" operation is known to be false without evaluating the righthand operand (Figure 2). Similarly, if a true value occurs on the left of an "or", the system can save time by not evaluating the possibly complex expression on the right of the "or". IRIS goes one step further. IRIS keeps track of whether the lefthand side of an "and" or "or" operator does a better job of predicting the result of that operator than the righthand side. If the righthand side is a better predictor, then IRIS will swap the two operands and evaluate the more efficient alternative first (Figure 3).

As the system scans through the data base in response to a query, it keeps a recent history of how effective a predictor each branch in the expression is. In one part of the data base, the lefthand branch may be the most effective. In another section, the righthand branch may be the most effective. IRIS constantly adjusts itself on a millisecond to millisecond basis to take best advantage of the prevailing prediction patterns.

In all other data management systems, the cost of evaluating a request increases linearly with expression complexity. In IRIS, the query tuning algorithm is so effective that the cost for asking a very complex request is the same as the cost for asking a small request containing only five or six Boolean operators (Figure 4).

2.3 Extension of the IRIS example

General purpose data management systems, as a rule, tend to be far less efficient than data management systems programmed for a specific application. For example, the general purpose data management system may have a tree structure capability that works to arbitrary depth. For a particular application only two levels may actually be used in the tree structure. The data management system tuned to this application would take advantage of that and not waste additional storage and processing to handle unnecessary pointers. The general purpose system will tend to be burdened with extra data fields and pointers.
Dynamic Expression Tuning

Expression Complexity (in operands, operators, etc.)

Cost for Three Evaluation Schemes

figure 3

figure 4
Our experience with IRIS indicates that the general purpose data management system does not have to be handicapped by its generality. The human programmer would notice that the data base in the previous example was only two levels deep. It is also perfectly feasible for the general purpose data management system to observe its own operation, discover this fact and restructure itself to be more efficient. The data management system does have the capability to analyze its operation quickly and in great detail. It should be able to respond more accurately and more rapidly than a human analyst and therefore tune itself better.

Of critical concern in such a self-tuning system is that the cost to measure all the parameters and to generate a theoretically optimum data structure or processing algorithm may in fact exceed the savings. For each dynamic tuning technique it is feasible to measure only a few parameters and to evaluate only small tuning algorithms. More often than not, the pragmatic optimum therefore differs from the theoretical optimum point at which data expressions or data structures can be reworked.

2.4 Increasing data utilization

The average data management system wastes a great deal of its time waiting for data to be read in. Data compression techniques can significantly reduce this time. However, even if data compression techniques are used, only a small fraction of the data read in is actually utilized in responding to a typical request. The cost of initiating I/O operations is so high that data records are usually blocked. However, when a block of 20 or 30 records are read, it is common that only 2 or 3 of those records are used. Furthermore, of the 2 or 3 records actually referenced, only 2 or 3 of the many fields in those records may be used. Thus, it is common to actually only process one or two percent of the data read in from a particular file block. If by appropriately restructuring the file, the number of records used per block and the number of fields used per record could be significantly increased, then the user would benefit. He would again decrease the
total amount of data read to get to useful data. He would also decrease the overhead to initiate I/O requests on many different file blocks. This will reduce both system load and response time. Perhaps a good measure of self-tuning system efficiency would be its data utilization rate.

Systems that have high data utilization rates are obviously more favorable in network environments. In a network environment the I/O rate between host sites is much slower than it is between any one host and its local peripheral devices. Therefore, some extreme measures to reduce the amount of data which must be transmitted to answer a given request would seem to be in order.
3. **Some Data Management Research Areas for WWMCCS**

Nine areas for suggested research are described. The first two, automatic compression and query tuning have relevance to any large file data management problem. The other seven, while valuable to data management in general, are particularly important to solving the problems of network data management with distributed data bases.

The data clustering and restructuring concepts are basic to the later areas of back-up, recovery, load leveling, etc. Little work has been done in the area of clustering and self restructuring systems. Experiments and prototypes which successfully attack these areas could form a technology base on top of which the problems of failure recovery and resource sharing become economically solvable.

3.1 **Automatic compression**

Several automatic compression techniques should be examined in terms of their applicability to the WWMCCS problem.

The first algorithm is a simple encoding scheme that computes the number of values permitted for a data field and provides the minimum number of bits that contain that value (minimum number of bits = \( \log_2 N \)). For example, a DOD payroll record might have a 24 byte last name field. If we wish to accommodate 16 million employees, we could assign a unique number to each employee and store this in a 24-bit field; this would be an 8:1 compression factor. However, we observe that most last names in a 16 million person file will occur many times. By taking advantage of repetition it would probably be feasible to accommodate all combinations with a 20-bit field.

The second algorithm is a minor modification of the first. A variable length field is introduced. For example, assume that 40 percent of the names in the payroll file are common names from a list of 1,000 common names. Use a 10-bit code to store one of the common names and a full 20-bit code to store any other name. This would mean that 40 percent of the data fields would be only 10 bits in
length and 60 percent of the fields would be 20 bits in length. The average field would be 16 bits long but a 17th bit would have to be added to indicate whether we are looking at a long or a short field. The overall reduction from the original 24 byte field to the 17 bit average field is better than 11 to 1.

The value of reduced data base size in WWMCCS is threefold. Storage requirements are reduced, response time is reduced, and network transfer time is reduced.

3.2 Query tuning

The concept of query tuning can be pushed much farther than it has been in IRIS. For example, the determination of the most efficient branch to evaluate next depends not only on the predicting capability of the branch but also on the cost. For example, assume one branch of an operator contains a small expression that costs ten (on some scale) to evaluate but has only 50 percent probability of predicting the operator outcome. Its opposing branch has 75 percent probability of generating a predicting value but requires the evaluation of a very complicated subexpression which costs 100 (again on some scale of dollars, cpu time, etc.). Since the most predictive branch is so expensive to evaluate, the low cost alternative is to evaluate the less predictive branch first.

The current IRIS system calculates approximate branch cost in the incremental compiler that prepares a query for execution. Since the dynamic tuning algorithm is very effective at reducing the cost of expression evaluation, the real cost of branch evaluation will be much less than the compiler calculated cost. IRIS would be even more effective if the cost of each branch were dynamically calculated like the prediction probability.

Currently, all IRIS Boolean operators are binary. We could do a better job of tuning the query if we were permitted to have an n-way "and" along with an n-way "or" operator. Each of the operands in such an operator could be ordered in terms of its cost/prediction capabilities. A highly predicting operand that may be
buried deep in a subexpression could then dynamically float to the top of the expression and predict the result very early.

3.3 Clustering

By observing access patterns on a data base it should be possible to determine what records in that data base are frequently accessed together. Similarly it should be possible to determine what fields in a record are accessed together. Those fields and records which commonly occur together are called a data cluster.

By extracting those fields and records from a conventionally structured file and putting them in one file block, the utilization of that block should rise significantly. In fact, the probability will be high that a request which accesses one field in a block will also want to access the other fields in that block. Similarly, if any record is accessed in a clustered block, access will probably also be required to its neighbor records.

A simple example would be a rectangular medical data base (Figure 5). In this data base, all of the data for a given disease occurs horizontally, across a row, and all of the data for a single patient occurs vertically, down a column.

As doctors use this data base, access patterns will emerge. For instance, one clinic of pediatricians will tend to access only children. Furthermore, those children will tend to have certain classes of disease like chicken pox and mumps and would not tend to have, for example, heart problems. Other doctors treating geriatric patients will more frequently access heart problems than chicken pox in their patients. The usage patterns of the physicians would indicate a clustering along the patient dimension and also a clustering along the disease dimension.

The determination of clusters is a difficult mathematical problem in combinatorics and graph theory. A lot of work has been done in the mathematics of the problem. However, it is normally required that each element be a member of only one cluster. The data clustering problem is a simpler mathematical problem because we are permitted to make copies of parts of the data base for use in more than one
PATIENTS

DISEASES

Pediatric patients

Geriatric patients

Childhood diseases

Heart diseases

Unclustered Data Base

Figure 5a

Clustered Data Base

Figure 5b
cluster. For instance, in our previous example, there is no reason to insist that a patient's record should only be in one cluster. It may be desirable to copy it two or three times. By copying the data into several clusters, sufficient processing and data transfer savings may be incurred to offset the additional storage cost.

3.4 Restructuring

In the previous example we are able to determine clusters in the rectangular data base. Since the access we have described so far has been patient-by-patient for the M.D.s it would seem reasonable to want the records of the file to be layed out vertically. Thus, all of the data for a given patient could be accessed with a single read. If, however, we also had medical researchers accessing the same data base and examining disease information, they would tend to access it in a horizontal fashion. The researcher would be best served if the file records were horizontal. By measuring the access patterns to the data base, it is possible to determine what is the optimal way to structure each cluster. Some clusters will be more frequently accessed by medical researchers and should be horizontally structured. Some will be more frequently accessed by the physician and should be vertically structured. Some will be frequently accessed by both parties and it would be cost effective to make two copies of the cluster—one stored horizontally and one stored vertically. Finally, there will be some clusters that are almost empty (e.g., heart diseases in children). It is not cost effective to store those small clusters either horizontally or vertically. A simple list of patient names and relevant disease observations would be the most efficient means of structuring the nearly empty cluster.

By making the data cluster small, i.e. a few thousand or tens of thousands of bytes, it becomes straight forward and relatively inexpensive to optimize the data structure for any given data base. The data structuring algorithm can be a dynamic one that operates on a second-to-second basis. This is important in WWMCCS applications. In a crisis situation, as bottlenecks occur, the data system measurement algorithms can identify the bottlenecks and immediately begin restructuring the
As the crisis deepens and access patterns become more pronounced, the data management system will tend to perform better rather than worse.

3.5 **Back-up and recovery**

If we have a clustered and dynamically restructured data base, the back-up and recovery problem looks more tractable. We propose that there should be a standard software module called the data management module at each site in the network that processes or stores distributed data. The data management module obeys a small list of 20 to 30 instructions. These are high level data management instructions that are capable, for instance, of creating a complicated key in a single instruction. We further propose that one or more identical log files be kept on a cluster-by-cluster basis. The log file identifies all operations that have modified the data cluster and the time at which that operation was issued. Each data base is also time stamped with the issue time of the last data modification request that was successfully executed on it.

In the event of failure of a primary copy of a data cluster it appears to be a relatively straightforward operation to read through the log backwards until we get to the time stamp that corresponds to the current data cluster. In the process of reading through the log backwards we are able to remove some superfluous and redundant operations. Once those operations are removed we will execute the commands contained in the log in a forward fashion sequentially. When we get to the end of the command list the data cluster is up to date and recovery is complete.

3.6 **Load Leveling and data base distribution**

It is feasible to have more than one active copy of a data cluster. Requests that do not modify a data cluster, but only read it, can go to any active copy of that data cluster. If this is combined with a status reporting protocol that allows hosts to indicate their load level and response capability, it will be possible to choose the least busy host, who has an acceptable copy of the data cluster, to execute the query.
Some requests do not require access to the most recent data. Slightly out-of-date data, perhaps a few minutes or a few hours old, will often be acceptable. Those requests could be routed to an older copy of a data cluster where a recent update had only been logged and not yet executed. Furthermore, since there may be multiple active copies of a given data base it seems reasonable to allow each cluster to have a different structure. In our previous example (sections 3.3 and 3.4) we might have had one cluster that was more frequently accessed by doctors than researchers. If three copies of that cluster existed, then two could be vertically structured and one horizontally structured. Thus, back-up copies are more than dead weight to be used only in event of failure. They can also be used to enhance performance and load level.

Since all hits on a data cluster are logged, it is easy to identify the most often updated clusters. Those clusters are the more volatile clusters in the data base, and they are more expensive to bring up to date in a failure recovery situation. The more time it takes to update the cluster, the more time that cluster is unavailable.

Consider an example. Assume 40 percent of the volatile data in the network system is on one machine and it will cost 40 minutes to bring all of that data up to date from back-up copies should that machine fail. It would be more reasonable to evenly distribute the volatile clusters across all machines in the network (possibly weighted by the probability of individual site failure). Each of 20 machines might have 5 percent of the volatile data in the system. This means, in the worst case, that only 5 percent of the volatile data could be lost in a single machine failure. One-eighth of the amount of data for the previous example would be inaccessible. Furthermore, since less update is required, it will only be inaccessible for about one-eighth of the time.

Clustering the data and recording command logs on the data by clusters has an interesting side benefit. In the event of machine failure only those clusters of
A data base actually being updated will be locked out to users. Once each cluster has been updated it will be immediately available for use even though other clusters are awaiting update. Thus, the vast majority of a data base should normally still be usable in a failure situation. In a distributed file system, as opposed to a distributed data management system capable of recognizing clusters, we would be forced to lock an entire file and prevent access to it even if only a small part were being updated. In the WWMCCS environment this could be a catastrophe, if that file were, for example, a critical status of forces file.

3.7 Data representation

If a consistent economical form could be found for expressing data structures and operations on data, it would make the analysis, measurement and tuning of the distributed data management system more straightforward (and in some cases feasible). We already know that there are more flexible data structures than the hierarchical tree commonly used in general purpose data management systems.

Codd relational form is a potential candidate for an economical data representation. This relational form is the subject of intensive data management research currently. The relational form looks deceptively simple and is based on expressing all data relationships as simple rectangular tables. The scheme is capable of generating all possible data structures and all possible data operations can be implemented on top of it. It has a very interesting attribute in that it would probably be easier to explain to a user than conventional tree structures. People are used to dealing with tables. Computer scientists are used to dealing with trees. Based on our experience with users, we think the table description would be more acceptable to a non-computer science user community and would be at least as general as, for example, an IDS file.

A major drawback to Codd relational form is that it requires a rather massive storage investment when implemented in a straightforward fashion. However, dynamic tuning looks like it may be feasible with a Codd relational data base. Dynamic tuning would probably remove the storage objection.
8.8 **Low error update algorithms**

Due to occasional errors in hardware or software it is possible for a spurious error to contaminate a data cluster. That error can then propagate via various update algorithms and remain in the cluster. The system will think that all copies of that cluster are identical, but in fact, one is different.

Techniques should be examined for putting low cost error detection and correction codes into data clusters and meshing these with data compression and update techniques.

8.9 **Data management and display in an intelligent terminal**

Graphics are very valuable in a report generator. It is an easy job for a smart terminal (that is, one with a small embedded processor) to prepare graphics locally but relatively expensive externally (in terms of processor requirements on a main host and communication requirements on the network).

Intelligent terminals should be researched in terms of their ability to provide data management and display capability. They can allow significant human engineering at the terminal. For example, a user need only log into his terminal and it could arrange to automatically dial into a communications front end. If that communications front end should fail, it could automatically dial an alternative. If adequate protocols exist, it should be feasible for the terminal to reconnect to a host and restart the terminal session with a minimum of interruption to the user.

Intelligent terminals have protocol implications. They can interact with the communications front end or a host for handling connections. If the communications front end contains an NCP, an intelligent terminal could act as a minihost. Intelligent terminals are also fully capable of handling their own terminal-to-terminal message traffic without imposing any load on or connection to a large host.

A multi-level data management protocol should be investigated that recognizes the limited, but potentially valuable, capabilities of the intelligent terminal. For
example, the intelligent terminal can handle ciphering and deciphering of extremely secure data bases at the terminal and never require that plain text be stored anywhere in the network system.

Intelligent terminals are already inexpensive and likely to get more so. There are processing units available on a single IC chip (e.g., the INTEL 8008 and 8080). These cost only a few hundred dollars. For a few hundred dollars more, a small amount of memory can be purchased. This means that for much less than a thousand dollars, a significant degree of intelligence can be added to a standard CRT or hard copy terminal. The cost of the processor and memory chip could be justified without requiring sophisticated applications (audio response, voice recognition, touch panels, graphics, etc.). For example, reduced host editing cost and message handling on the network are probably sufficient cost justification.
4. **WWDMS Enhancements**

The current WWDMS is a single site data management system exploiting conventional data management technology. A WWDMS enhancement program is needed to transform the current WWDMS into a self-tuning and fully networked WWDMS. An R & D program to develop the technologies described in section 3 is assumed. We address here the problem of adding proven data management technology to WWDMS (e.g., data compression and query tuning) and immediately exploiting new network data management technology as it is developed (e.g., clustering and load leveling).

4.1 **Subsystem command**

The current COBOL based file structure of WWDMS is incompatible with the proposed compression and query tuning facilities. These facilities require that data fields be based on bit rather than byte boundaries and that measurement facilities be added to files and commands. WWDMS must be modified to be able to exploit these new technologies.

One interactive command could be added to WWDMS. This command would enter a compression and self-tuning data management subsystem. The subsystem could initially contain compression facilities. Query tuning facilities would be added. As networking concepts were proven in the research program, they could be added to the subsystem.

4.2 **Upward compatibility**

At all times the subsystem facilities would remain compatible with the current WWDMS commands. Bridge facilities would be implemented in the subsystem to transform a conventional WWDMS file into a compressed and tuned file and to transform the subsystem files back into conventional WWDMS files that may be accessed by COBOL applications programs.

The most frequently used data operations would be available within the subsystem. Infrequently used operations, sophisticated report generation, and
special applications would continue to be performed with the current WWDMS facilities and external COBOL programs.

Like most general purpose data management systems, WWDMS is scheduled to have several hundred capabilities. Yet experience tells us that only a small fraction of a total system is normally used. The common phrase is "ten percent of the system is used ninety percent of the time and ninety percent is used ten percent of the time". By limiting the subsystem to that part of WWDMS used most heavily, we can

1. Keep the subsystem smaller,
2. significantly reduce the cost of implementation,
3. make the subsystem available sooner, and
4. gain the advantage of improved performance and reliability for the most common/critical WWDMS tasks.

4.3 An implementation plan

As an example of how these enhancements could be added to WWDMS, we have prepared a simple, four phased scenario. The four phases provide for an orderly transformation from the current single-site WWDMS into a fully networked WWDMS which can accommodate the new technologies discussed in section 3.

4.3.1 Phase I - subsystem & compress

The first phase of the enhancement program will implement the subsystem with the following capabilities:

1. file creation
2. file deletion
3. read a standard WWDMS file into internal compressed format
4. write an internal file out in standard WWDMS format
5. a minimum set of interactive update and query commands
6. hooks for adding instrumentation and measurement.
4.3.2 Phase II - tune & measure

The second phase of enhancement will add the following subsystem capabilities:

1. query tuning
2. an instrumentation and measuring package
3. user abbreviations
4. on-line help (TUTOR command in current WWDMMS)
5. additional interactive commands requested by the user community.

4.3.3 Phase III - clusters & logging

Once phase II is complete, a basic subsystem capability will be available as a single site service. Phase III additions will prepare the single site system for networked operations. These additions will be based heavily on concepts proven and pitfalls discovered in a research program addressed to the problems discussed in section 3.

Examples of phase III capabilities that might be added are:

1. a clustered data base
2. a clustering measurement package
3. a logging package for each cluster
4. a user command to manually examine clustering alternatives
5. a user command to manually force cluster restructuring and possibly recompression

4.3.4 Phase IV - network operations

In phase IV, distributed operations would begin. Examples of likely phase IV enhancements are:

1. automatic clustering
2. automatic restructuring
3. resilient data management protocol
4. automatic back-up
5. load leveling
5. Conclusions

The concepts and programs discussed in Section 2, 3, and 4 of this paper all have direct impact on WWMCCS performance and survivability. Some of these concepts have already been proven in production systems and would benefit from further research (e.g., data compression and query tuning). Others have not yet been attempted and are in need of research programs to develop their potential (e.g., clustering, restructuring, load leveling, etc.). One plan has been described to transform the current WWDMS facility into a far more powerful and responsive network tool. While the proposed scenario may not be the best approach, it does demonstrate the feasibility of exploiting these radical new technologies in a way that is compatible with existing and previous programs.
6. Bibliography


42. M. Heine, "Distance Between Sets as an Objective Measure of Retrieval Effectiveness", Information Storage and Retrieval, Vol. 9 (March 1973), pp. 181-198.


78. J. Palme, "Making Computers Understand Natural Language", in Artificial Intelligence and Heuristic Programming, pp. 199-244.


