This paper will appear in the Proceeding of the ACM SIGIR 1987 International Conference on Research and Development in Information Retrieval, under the title "Data Caching in Information Retrieval Systems."

THE APPLICATION OF WORKSTATION CACHING TO INFORMATION SYSTEMS

Patricia Simpson
Rafael Alonso

CS-TR-085-87

March 1987

THE APPLICATION OF WORKSTATION CACHING TO INFORMATION SYSTEMS

Patricia Simpson and Rafael Alonso

Department of Computer Science
Princeton University
Princeton, New Jersey 08544

ABSTRACT

Information retrieval (IR) systems provide individual remote access to centrally managed data. The current proliferation of personal computer systems, as well as advances in storage and communication technology, have created new possibilities for designing information systems which are easily accessible, economical, and responsive to user needs. This paper outlines methods of integrating personal computers (PCs) into large information systems, with emphasis on effective use of the storage and processing capabilities of these computers. In particular we discuss means for caching retrieved data at PC-equipped user sites, noting that caching in this environment poses unique problems. An event-driven simulation program is described which models information system operation. This simulator is being used to examine caching strategies. Our studies show that the limiting factors in caching effectiveness are the speed of the communication channel and the proportion of session time during which that channel is idle. When these values are low, unrestricted caching can seriously degrade performance. However, when they are sufficiently high, caching becomes a viable means of distributing data to users.

THE APPLICATION OF WORKSTATION CACHING TO INFORMATION SYSTEMS

Patricia Simpson and Rafael Alonso

Department of Computer Science
Princeton University
Princeton, New Jersey 08544

1. INTRODUCTION

Information systems constitute one of the oldest applications of digital computers, and one whose relevance and importance increase with time. In part because of their age and established nature, information systems today operate in much the same way as they did twenty years ago. Yet certain technological advances have been made in recent years which, if carefully integrated with modern systems, could substantially improve their availability, economy of operation, and ease of use.

Perhaps the most significant of these advances is the personal computer or workstation, part of a larger evolution toward inexpensive, small-scale, distributed computation. Along with microcomputers have emerged personal, portable storage media, primarily disks and diskettes. Add to these a widening choice of digital communication media, and exciting possibilities begin to appear. We believe that these elements together have the potential not only for improving the operation of existing information systems, but also for dramatically expanding their role in society.

1.1 Characteristics of information systems

Although many varieties of information system exist, ranging from broadcast videotex services to home shopping and banking, we will in this paper assume a definition which is characteristic of the mainstream of current systems. First, we assume that the service and its users are independent of each other. This leaves out management information systems and others which are internal to an organization. Most users connect to the system over telephone lines with the aid of modems.

Secondly, the service takes all responsibility for acquiring, organizing, and updating the information it provides, and for defining the means of access. Users only read the data, having no other control over it. Thirdly we assume that interaction is in the form of a short query followed by a fixed response from the system. Users do not submit programs or complex queries of a procedural nature. Most queries consist of a single command with operands, or an indication of a choice from a menu. Very few are longer than one input line on a terminal.

Lastly, the output is in display format (e.g. ASCII), intended to be read directly by the user at a terminal. This is because the typical information service presumes that its users operate terminals (not computers) and have no use for machine-readable data. Many systems limit response length to the capacity of a single screen for the same reason. This lowest-common-denominator approach by information services, while easy to implement and maintain, frustrates many users as it severely limits their means of interaction with a system that is potentially much more useful.

1.2 Adding desktop computers to IR systems

The desktop microcomputer has become widely available in recent years, to the point where the great majority of subscribers to online information services use personal computers rather than terminals to access them [Gold85]. At present PC-based users are scarcely better off than users of so-called dumb terminals; the special capabilities of personal computers — storage, computational power, and potentially high-level user interfaces — have not yet been put to work in the information system environment. Information services today operate in a highly centralized manner, defining a single interface for all users. This inflexibility, together with the substantial expense of interactive telecommunication, causes most PC users (an estimated

80-85% of them [Dunn84]) never to use any information system at all. Personal computers offer a remedy for this unhappy situation. An information system, generally viewed as a monolithic mainframe-based database to which users occasionally "attach", can instead take on the role of auxiliary processor or server, providing data on demand to supplement the otherwise autonomous operation of personal workstations. It is now possible to place each user at the center of the system, rather than at the periphery, and to tailor the user's view to suit individual needs. It is our claim that this decentralization is the next logical evolutionary stage for information systems; a new dimension of usefulness can be achieved through recognition and use of small computer capabilities. Among the benefits that can be expected are:

Greater autonomy for the user. When users can download information and manipulate it locally, they have more control over their own use of it. Every user can build a personal database, gleaned from many sources, containing the information of interest to him or her — much as people now assemble personal libraries of printed material over time. Each user can employ private software to process the data in ways not provided by the information service. This makes the service much more valuable to its subscribers. Availability of the system is also increased in the sense that, even when the host site is down, it is possible for users to have the data and programs they need available locally.

Better utilization of resources. If some processing and storage responsibilities can be delegated to user computers, there will be less contention for resources at the host site. This should result in quicker system response overall, as well as the ability of the service to accommodate more users at one time. Since personal computers are single-user machines, their processors are typically idle during online sessions; they therefore constitute a source of "free" processing power.

Lower communication costs. Communication between central site and user is an expensive part of system operation. Our studies show that a high percentage of online session time is spent in data transmission, a result of using standard telephone lines which necessitate slow transmission speeds. The more work that the user is able to do locally, the less need there is to communicate with the host system. For example, consecutive queries accessing the same data should be processable locally, rather than through the central site each time.

Improved user interface. Query languages can be confusing and difficult to use, especially for users who need to work with many different systems. Studies of interactive information systems have found that a surprisingly large number of queries contain errors and must be redone [Borg83,Penn81]. A PC can be programmed to help users formulate queries, checking for validity and syntactic correctness, providing menus of command choices, displaying subject term thesauri, and so on. It can do this within the environment to which the user is accustomed, using windows, menus, a light pen or mouse, and other tools to make interaction easier. Perhaps most importantly, each user can have a different interface, customized to meet individual needs. This is impossible within the mainframe/terminal paradigm, one reason why today's systems are underused.

Increased functionality. Many valuable enhancements to IR systems have been proposed, but are seldom implemented in practice because of the great expense and effort required to make significant changes to an already-working system. Some of these functions may be performed by user computers with little or no centralized effort. Current-awareness service is a simple example. A user computer can maintain locally a profile of subject areas of interest, and automatically query the service periodically to update the user's personal database in those subjects. Another facility might permit a user to weight the keywords used in a query according to their importance; the PC can then post-process the responses received, assigning a relevance value to each and sorting them by decreasing relevance for the user's perusal.

2. RELATED WORK

Several promising attempts have been made at giving small computers a place in IR systems. Jamieson's intelligent terminal [Jami79], built before PCs were commercially available, was designed to implement and test sophisticated retrieval strategies without modifying the operation of the central system. More recently, commercial information services have begun to provide the means for users to download data and manipulate it locally. IRS, a European service, has a DOWNLOAD facility which supplies bibliographic citations in a format that can be easily processed by user software [King86]. The users themselves must supply such software. Dow Jones News/Retrieval, on the other hand, sells PC software to capture data from its financial databases and produce graphic output [Glos83]. While a step in the right direction, the software applies to only a small subset of the system's information and performs a very limited set of operations on it. Users still have little control over their use of the data.

An interesting project is the Boston Community Information System (BCIS), a pilot service developed at M.I.T. [Giff85]. This system broadcasts information of community interest (such as current news) over FM radio to IBM PCs which have been provided with data management software. The entire database is transmitted repeatedly, in cyclic fashion. The PC software filters incoming data, storing only the files which match criteria previously specified by the user. When the user needs access to the system's information, the files of interest are waiting. In this way the selection process is completely offloaded to each user site. A drawback of a pure broadcast system is that the database to which a user has access at any given time is limited to the size of available local storage. Access of an unstored file must wait until that file is next broadcast. To remedy this problem, BCIS will provide dialup access so that users may solicit and receive specific data files when needed.

Our work differs from this approach in a few important ways. We place no restrictions on the size or content of the database; it may be quite large and diverse, containing data that changes rapidly (perhaps many times per day) in addition to longer-lived data which might remain fixed for years. Some items will be of interest to many users, but most will only rarely be accessed. Volatile data, to be useful, must be retrievable on demand and rapidly replaceable. Cyclic broadcast of data cannot accommodate this need for direct interaction, nor can it distribute very large databases in a timely manner. We are also interested in automating the decisions of what data is to be downloaded and when, rather than requiring that users specify those choices in advance. Ideally, decision-making should adapt dynamically to each user's changing access patterns.

We would like to develop mechanisms for distributing data to users which are applicable to a wide variety of systems, handling arbitrary data and not restricted to workstations of a particular make. This is an ambitious goal, given the unstandardized state of personal computer systems. It therefore seems wise to begin with general, straightforward methods that can be put into operation quickly without major restructuring of system operation.

3. DATA CACHING

Downloading to common storage devices as part of an ordinary online session is probably the simplest way to provide data, requiring the least modification of the way current systems operate. No novel communication or storage media are required, and software changes at the central site are minimal.

Because a user is concerned with only a small portion of the available information, we must be selective about what is downloaded, and when. We expect the greatest benefit from local storage when the data downloaded will be accessed many times in the future before it becomes obsolete. While it is impossible to predict future use, the technique of *caching* has proven valuable in other environments when decisions about data placement must be made [Smit82]. Caching appears promising here as well. By caching we mean downloading data *as it is accessed* into the user's available local storage. A rudimentary form of caching would simply store the output from a query for reviewing. A more sophisticated (and useful) version would cache whatever data items are necessary to re-execute locally the query which was just processed at the central site, and presumably other queries on that data as well.

As in a CPU cache, data is brought physically closer to where it is actually used. This is done in the expectation that the same data will be referenced again in the near future. As with standard caching, appropriate replacement policies are needed for efficient operation, and some mechanism must exist for recognizing when cached data has become out-of-date and so must be reacquired.

Although well understood in the context of hardware memory caches, caching in the information system environment has not been fully studied. There are several significant differences between standard memory caching and the kind we are presenting here. First, since users do not perform updates, there is no need to keep track of changes and write modified data items back to the central site. When a cached item is to be replaced by another, it can simply be discarded (overwritten).

On the other hand, keeping a cache up to date is much more difficult because updates originate at the central site, not at the site holding the cache. Solutions to the multiple-cache problem within multiprocessing systems do not apply, for two reasons. There is no continuous connection over which updates might be propagated as they occur, since users are free to disconnect from the service whenever they like, and it is clearly impossible for the service to know which users have which versions of which data so that it might transmit updates to each at their next session. In addition, user needs will vary; some may want to receive updates automatically, but others may not want updates at all or may want to receive them only when they specifically request them or only for certain data items.

There is also the issue of conversion. When moving data between two very different systems it is likely that conversions in coding scheme, word length, numeric precision, character set, file organization, and/or structure of addresses, indexes, and pointers must be made. Software at the user site must understand the format of cached data and be capable of accessing and manipulating it.

There is a choice of storage areas for a cache. The most obvious are main memory, disks, and diskettes, although high-density write-once optical disks can be expected to gain acceptance in the near future for storage of data of long-term value [Rose83]. Among these various media there are tradeoffs between capacity, convenience, speed, and cost.

The items being cached are not of a fixed, small size. They may range from a few words to many hundreds of kilobytes. The question arises of how best to organize them in local storage for subsequent retrieval. Varying size results in varying communication delays as well. When large items are to be transmitted we must ensure that normal interactive use of the system is not greatly impeded.

An important consideration is that normal query output is not necessarily suitable for caching. Output is provided in display format for immediate viewing, not for machine storage or manipulation. It is clear that the caching of a record or file will involve an extra transmission of "raw" data in addition to the screen-oriented response to the user's query. For example, a request for the average price of a stock over the previous seven days normally returns a single value, coded in ASCII and displayed on a screen. It is desirable to cache the price for each day as a separate machine-readable value, so that, for instance, the average price over the previous five days can be computed locally. Thus the action of caching is not "free" as it is in standard single-system caching environments. Any caching mechanisms employed must perform well enough to offset this expense.

Because this type of caching is not subject to strict timing constraints as is hardware memory caching, much more complex decisions can be made during system operation concerning what should be cached and where, and what should be replaced when a cache is full. Indeed, the factors mentioned above make it imperative that sophisticated caching techniques be explored. Communication is a potential bottleneck in an information system and any form of caching must involve additional communication.

4. SIMULATION OF IR SYSTEMS

4.1 Simulator design

We have written an event-driven simulation program which is capable of modeling the operation of an information system. Our primary design goals were accuracy and flexibility. We have tried to keep the overall structure simple, yet with sufficient detail to ensure that all elements significantly affecting system operation are represented. We chose simulation rather than analytic modeling because it seemed unlikely that an analytic model could capture the complexity of interaction inherent in a large information system with its diversity of users. Moreover, it is important that the model be easily extended to test modifications to the system. The present version supports six levels of caching and three cache replacement policies. In the future we expect to simulate other caching strategies as well as data prefetching, broadcasting, and shared communication lines. It is hoped that this design is flexible enough to incorporate these additions easily.

The program was written with the aid of SIMPAS [Brya81], a tool for constructing event-driven simulators. SIMPAS generates most of the code for managing the event queue, advancing the clock, and gathering certain statistics. It also supplies routines for generating random numbers from several distributions. We found SIMPAS extremely helpful in the programming process as it eliminated many opportunities for error and enabled us to concentrate on higher-level design issues.

As shown in Figure 1, the basic entities in a simulated system are the host processor, its secondary storage, the users, their processors and secondary storage, and

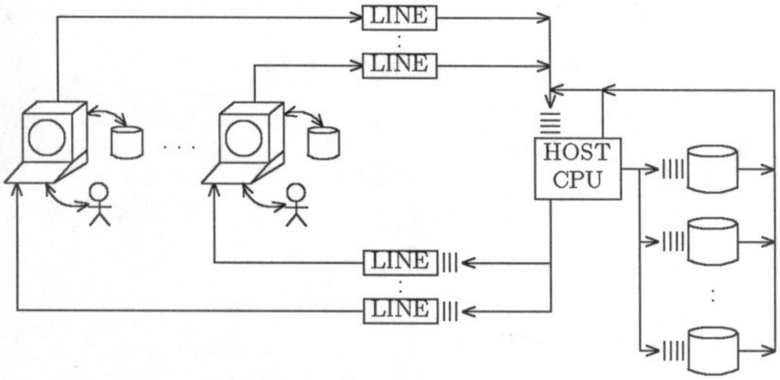

Figure 1: Model of information system

the communication lines. We will call these the *fixed* entities. Although the parameters of secondary storage devices can be defined to represent different media, we will generally refer to them as disks. Currently all communication lines are point-to-point lines serving one user each.

Queries are temporary entities which represent the interaction between the user and the information service. The user generates a query, or request, which travels over the communication line to the central site, is serviced alternately by the host CPU and disks (queueing behind other queries if necessary), and returns with the system's response via the communication line to the user. The movement of queries within the system is shown in Figure 1 by arrows. When data is to be cached, the returning query is accompanied by a second temporary entity representing that data. If the necessary data for a query has already been cached, then the query does not travel to the central site at all but is serviced instead by the local CPU and disks. The assumption is made that every PC possesses software to implement the full functionality of the central system. Upon receiving the output of a query, the user waits for some period of time and then initiates a new query. Each user is associated with exactly one query at a time.

4.2 Purpose

The simulator was designed to serve several purposes:

To provide a good model of current information system operation. Simulated users can operate terminals instead of computers simply by defining no processor speed or storage space for them. Such a model has been useful in "getting a feel for" the interaction between current (noncaching) systems and their users. By examining the interplay of such factors as line speed, user population, and database size, we can determine where performance bottlenecks are most apt to occur and thereby identify where the most productive enhancements to the basic system may be made.

To determine an upper limit on the usefulness of PC involvement. To find out how powerful desktop computers must be if they are to be useful as IR processors, we can simulate offloading all data and processing to them, varying their capacity and speed if necessary.

To measure the effects of data caching on system performance. We can model different degrees of caching and various replacement policies to determine the conditions under which caching improves response time and lessens central-site congestion, as well as those under which performance degrades.

4.3 Parameters and model validation

The results of any simulation experiments, of course, depend on the choices of input parameter values. Our simulator takes as input about 30 parameters. Acquiring reasonable, realistic values for some of these has been straightforward. With others, however, substantial uncertainties remain. The following sections discuss each parameter in detail, explaining these uncertainties as they arise, so that the reader may understand the operation of the program well enough to interpret its results with confidence.

4.3.1 User configuration parameters

We associate with each user a personal computer, the specification of which is called a user *configuration.* From one to eight different user configurations may be active during a given run. Disks are characterized by four parameters: capacity (in bytes), track size (in bytes), data transfer speed (in bytes per second), and overhead for each track access (in seconds) due to seek time and rotational delay. Values for these parameters are all easily obtained from literature describing peripheral storage devices for modern PCs. Per-track overhead time is a fixed value, equal to the average value for the device. We assume that each PC has only one disk. Multiple disks can be simulated by defining a greater disk capacity. Thus it is assumed that multiple disks are of the same type and speed, and that only one is active at a time. For our present purposes this is sufficiently realistic.

Communication line speed is expressed in bytes per second, and is identical in both directions (user to host CPU and host CPU to user). Typical values for telephone lines are 30 and 120 bytes per second. We assume no additional overhead time per transmission.

CPU speed is expressed in MIPS. This has proven to be a difficult parameter to define. What is wanted is some measure of the time required to perform a "standard" sequence of operations involving the retrieval and processing of data. This is a nebulous concept to begin with, since we do not have actual instruction sequences to examine and so cannot state exactly what takes place. In any case, instruction mix would vary greatly depending on the type of processing being done (searching, sorting, formatting output, etc.). Characterization of a typical mix is further complicated by the intrinsic differences between large and small computers. If a mainframe and a microcomputer had identical instruction sets, then MIPS alone would be sufficient to differentiate between the processing power of the two machines once an instruction mix had been established. But instruction set differences are so great that measurements of relative CPU speed are approximations at best. However, we do not expect processing speed to be a limiting factor in an IR system.

4.3.2 Application parameters

An *application* definition characterizes the type of access a user engages in, and incorporates quite a bit of flexibility. From one to eight applications may be defined for a given run, in any combination with the available user configurations. Statistics are gathered for each group of users which share the same configuration and application.

Think time is the average time (in seconds) that elapses while the user examines the output from a query and formulates a new one. This includes the time required to type the next request. Input size is the average number of bytes used to transmit a query to the host; it includes both what was typed and any necessary overhead bytes. Output size is the average number of bytes returned to the user in response to a query. Note that output size pertains only to the screen-oriented response; it does not include any machine-readable data which may be sent in conjunction with the standard response.

Typical values we have used for the think time, input size, and output size parameters are 10-60 seconds, 100 bytes, and 1000 bytes, respectively. Sample values in each case are drawn from a normal distribution with a standard deviation of $\mu/5.15$, which places 99% of the values between $\mu/2$ and $3\mu/2$. This choice of distribution and of the mean values themselves is primarily an educated guess arrived at by examining user manuals for two systems — Compu-Serve and Dow Jones News/Retrieval — and by using each system for a short period of time. Although we have no precise measurements for these values over a wide range of usage, it is clear that those chosen are certainly reasonable.

Next are specified the average quantity of data read from disk, and the average number of machine instructions executed, per query. Again we use a normal distribution as described above to choose the exact values for each query. The experiments reported on in this paper all assumed an average of 10,000 bytes accessed per query; this is roughly equivalent to 50 bibliographic citations or 3 printed pages of full text. The average number of instructions was fixed at 2.5 times the number of bytes; we cannot be sure that this is reasonable, but it is based on the assumption that information system activity is not computation-intensive. We hope that in the future we will have the opportunity to trace periods of use on an actual system to get a better idea of the ratio of computation to disk access.

Also part of an application definition is database size, or more specifically the size (in bytes) of whatever *portion* of the system's data might be accessed by a user of the application in question. All results presented here are based on 10 megabytes. This parameter reserves an appropriate number of central-site disk blocks for the application.

Locality of reference is defined by dividing an application's blocks into three groups and specifying what percentage of accesses are made within each group. We defined six levels of reference locality, based on the Bradford distribution [Broo69], which states that given some collection of journals arranged in decreasing order of the number of "interesting" articles contained therein, and then partitioned into k groups of journals each containing the same number of relevant articles, the number of journals in the groups fit the proportion $1{:}n{:}n^2{:}\cdots{:}n^{k-1}$. We have adapted this idea to a database environment by dropping the journal boundaries and specifying groups of blocks each containing the same number of relevant blocks (i.e., each group is accessed with equal frequency). Levels are identified by the Bradford multiplier (n) used; level 1 represents uniform access. Although the analogy is rough, it permits an organized way to specify increasing degrees of locality. The values used are shown in Figure 2.

Locality level	*Percent of data accessed 1/3 of the time*		
1	33 1/3	33 1/3	33 1/3
2	57	29	14
3	69	23	8
5	81	16	3
10	90	9	1
15	93.4	6.2	0.4

Figure 2: Increasing levels of reference locality

Lastly we assign a lifetime to the data blocks in each of the three groups. Cached blocks are timestamped upon arrival and are automatically deleted from the cache when they expire. It is not difficult to estimate good values for this parameter based on a knowledge of the types of information supplied by a system. Simulations need not address the far more difficult question of how a block's expected lifetime might be determined in a real system.

4.3.3 Central site definition

The central site consists of a single mainframe processor which services queries in round robin fashion. Each query alternates between executing instructions and reading disk blocks. If a query has not finished or requested disk service after 50 milliseconds of CPU use, it is interrupted and requeued at the end of the CPU queue.

CPU speed is specified in MIPS but, as mentioned earlier, this is only an approximate measure of real computing speed. It is important, however, that the *ratio* of mainframe speed to PC speed be accurately represented. The simulation runs reported upon in this paper used values of 4.0 and 0.5 MIPS for the mainframe and

PCs respectively.

The remaining central-site parameters describe secondary storage characteristics: capacity, blocksize, average seek time, average rotational delay, and data transfer speed. These are all straightforward. We have been using values representing IBM 3350 disks with a blocksize of 4096 bytes. Disk capacity is 300 megabytes (a convenient approximation to the actual value of 317.5 MB), average seek and latency times are 25 msecs and 8.33 msecs respectively, and transfer speed is 1.198 MB per second.

In general we do not intend to experiment with many different central site configurations, but it is convenient to parameterize these characteristics so that we can test whether reasonable variations in their values greatly affect our results.

4.3.4 Run parameters

The few remaining parameters define certain characteristics of the run as a whole. The length of the simulation is given in minutes, as is the time interval between successive printings of output statistics. Generally we run for 60 minutes and generate output every 10 minutes. This is reasonable because most real sessions are less than an hour in length.

Each run employs one of six caching options: cache every block accessed to local secondary storage, cache every block accessed to local main memory, cache to disk only blocks which have been requested more than once, cache to disk only long-lived blocks, assume that the disk cache initially contains *all* data blocks (all processing is local), or do no caching at all (all processing is central). All but the last two options require a replacement policy. One of three can be chosen: replace the oldest block (FIFO), replace the least recently accessed block (LRU), or stop caching new items until some cached block expires (replace only expired blocks).

From 0 to 900 users may be specified for each combination of a user configuration and an application (we call each such combination a "group"), provided that the total number does not exceed 900. Population not only affects the overall workload of the system, but large user groups serve to "even out" the output values obtained, since totals and averages are reported by group rather than by individual user.

Output values include: average and maximum queue sizes at the central CPU and disks, average and maximum response times and volume of data transmitted per query, and the average percentage of time spent during each session for processing, I/O, transmission, waiting in queues, and thinking.

5. RESULTS AND DISCUSSION

Performance measurement of an information system is a complex issue, encompassing not only such tangible phenomena as response time and queue lengths at shared resources, but also the more slippery concepts of suitability to the users' purposes, adaptability, and ease of use. Simulator output gives us clear insights into the quantifiable metrics; from these measurements we must infer the others. Cache hit ratio, for example, has a strong bearing on response time; but a high hit ratio also means that the user is more likely to be able to continue processing should the central site go down. The degree to which users "possess" the data and can use it to advantage is perhaps the most important performance criterion and, although it cannot be measured directly, we must keep it in mind as we examine more concrete results.

5.1 Effect of line speed on standard operation

For reference, we begin by measuring a "baseline" system of 400 users, each accessing at most 10 megabytes of a 1200-megabyte database contained on four IBM 3350-type disks at the central site. No caching is performed, and access of the database blocks is uniformly distributed. A request requires 100 bytes of input and elicits 1000 bytes of display output, on the average. Think time averages 10 seconds. Users differ only in their data transmission speed; 100 each transmit at 30, 120, 480, and 960 bytes per second. All users are active simultaneously for one hour.

Response time is good; even at 30 bytes per second it averages less than 4 seconds. Contention for CPU processing is low, with the CPU active only 15% of the time. Some disk contention is evident; each disk is busy about 65% of the time. Still, less than 1% of the duration of the average query is spent queued for disk service. Figure 3 shows response time for this no-caching, or standard, case. (Note that response time is graphed on a logarithmic scale throughout this paper.)

The primary bottleneck is in the transmission of queries and their results. 10%

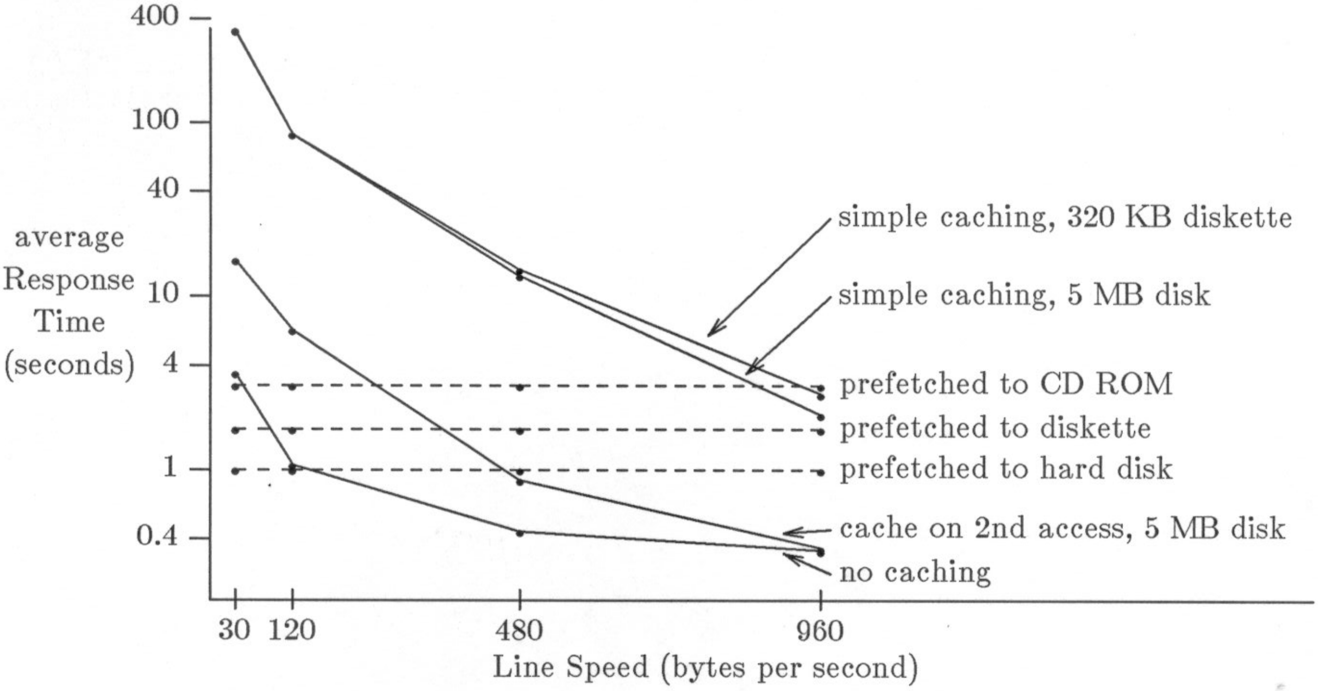

Figure 3: Caching and full prefetching compared to standard access, uniform reference

of session time is spent in data transmission at 960 bytes per second, 18% at 480, 47% at 120, and 78% at 30. Currently few information system users can transmit at higher rates than 1200 baud. In light of this it is clear that caching will carry with it substantial overhead.

5.2 The addition of simple caching

Next we provide each user with a single diskette which can hold a 320 KB cache, or about 3% of the database. Again, uniform access is assumed. We implement the simple policy that every block accessed is cached, and if all blocks necessary for a particular query have been cached then that query can be executed locally. We assume for now that no block expires — once it has been cached, its contents are valid for the remainder of the session. FIFO replacement is used. Each query now causes the transmission of 11,700 additional bytes on the average. This has a disastrous effect on response time, as seen in Figure 3. While at 960 bytes per second it remains under 3 seconds, it leaps to 14 seconds at 480, 86 at 120, and nearly 6 minutes at 30. Worse still, the average response time does not decrease over the course of the one-hour session; the cache is useless.

One reason for this behavior is that an average of three blocks are needed to execute a single query, *all* of which must be in the cache if the query is to execute locally. (In this context, "hit ratio" is the ratio of locally executed queries to all queries). Conjecturing that a single diskette is too small to serve as a cache for a database of this size, we increase the cache size to one 5-megabyte hard disk, or over 50% of the database. Yet this change produces nearly identical results. At 9600 baud, although response time drops from 2.74 to 2.08 seconds, the hit ratio reaches only 2½% after an hour of operation. At slow speeds no improvement is seen.

The primary problem is that it takes a very long time to fill the cache to a point where it becomes useful. It is interesting to note that even at 9600 baud it would require about 90 minutes of continuous transmission to fill a 5-megabyte disk. When the database is accessed randomly, virtually every query requires a large transmission of data to the cache, and this delays the transmission of the following query's response.

5.3 Effect of locality of reference on caching

Our assumption of uniformly distributed access is, for most applications, unnecessarily strict. When most of a user's access is within a small portion of the data base (a "hot spot"), caching must be more effective. We repeated our "simple caching" simulations (5 MB cache), but with increasingly localized access. At the higher line speeds hit ratio increases significantly as locality increases (Figure 5). After an hour's operation, 9600-baud users are processing over 25% of their queries locally. The slower lines, however, cannot build the cache quickly enough to have much effect. Even with very high locality response times are much worse than with no caching at all (Figure 4).

It is evident that caching must be restricted in some way if low-speed users are to experience acceptable response times. We applied a simple heuristic: cache blocks only on the second access, rather than the first. In this way time is not wasted caching rarely used blocks. This first cut at basing the caching strategy on actual patterns of use produced interesting results. The reduced transmission volume does improve response times substantially (although they are still unpleasantly high for

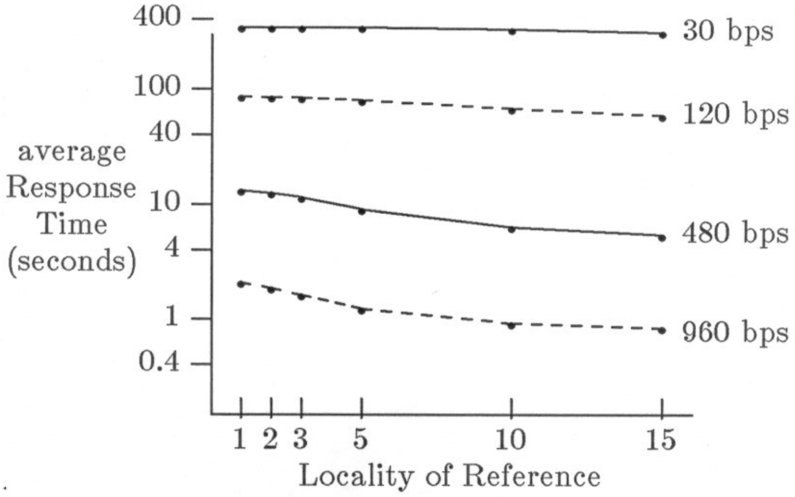

Figure 4: Simple caching (5 MB disk), with hot spots

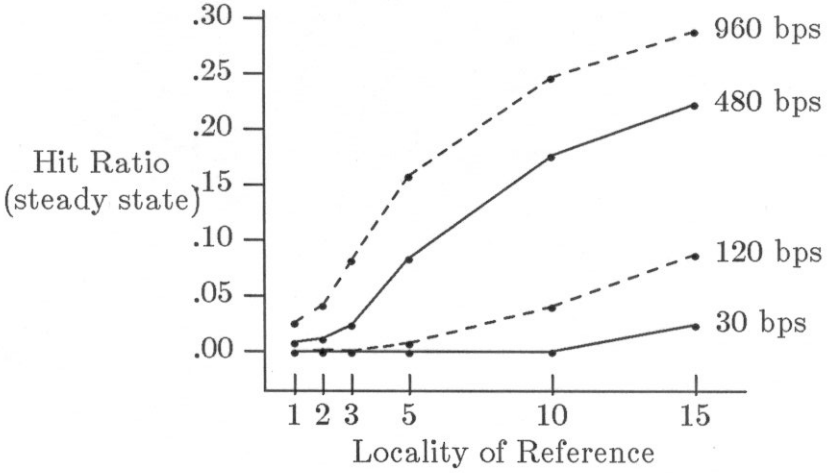

Figure 5: Simple caching (5 MB disk), with hot spots

low-speed users), but the improvement is greatest when locality is low (Figure 6), demonstrating again the extreme sensitivity of system operation to transmission time. The slower the line speed, the *greater value* a data item must have to make its downloading worthwhile. At high speeds hit ratios decreased; transmission lines were often idle when they might have been caching (Figure 7). At low speed the hit ratio increased; more queries were processed during the hour, and more high-value blocks were cached.

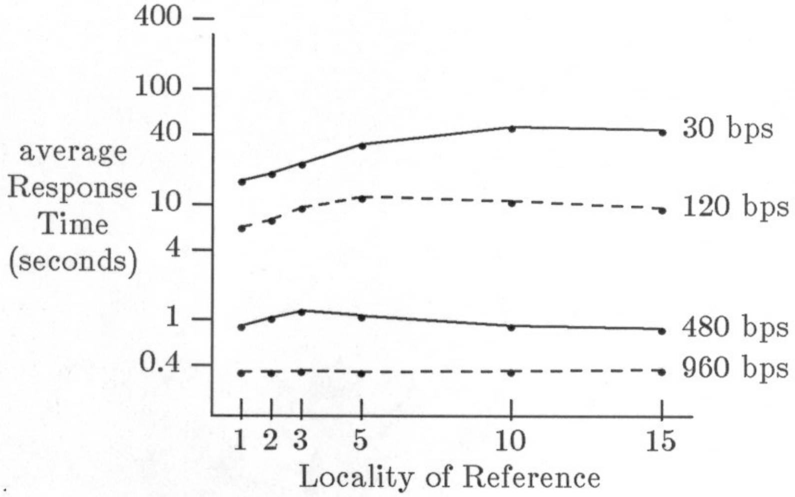

Figure 6: Caching on 2nd access (5 MB disk), with hot spots

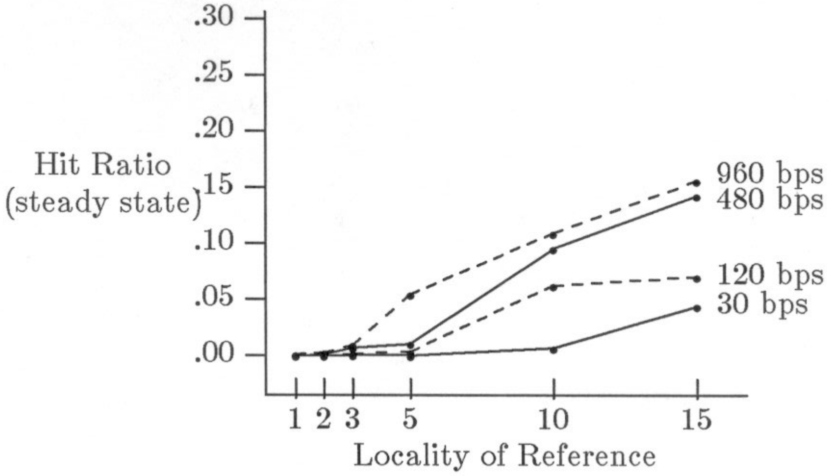

Figure 7: Caching on 2nd access (5 MB disk), with hot spots

5.4 Effect of think time

Recall that the foregoing simulations assumed an average think time of 10 seconds. Although suitable for some applications, this value is quite low for others. We examined the effects of longer think times. Interestingly, this "idle" time during the session can be crucial to caching performance. Increasing think time lessens the demand for data transmission while providing more time during which to transmit. Figure 8 (solid lines) shows the resulting decrease in average response time. For comparison, the no-caching case is also shown (dotted lines) for think times of 10 and 100 seconds.

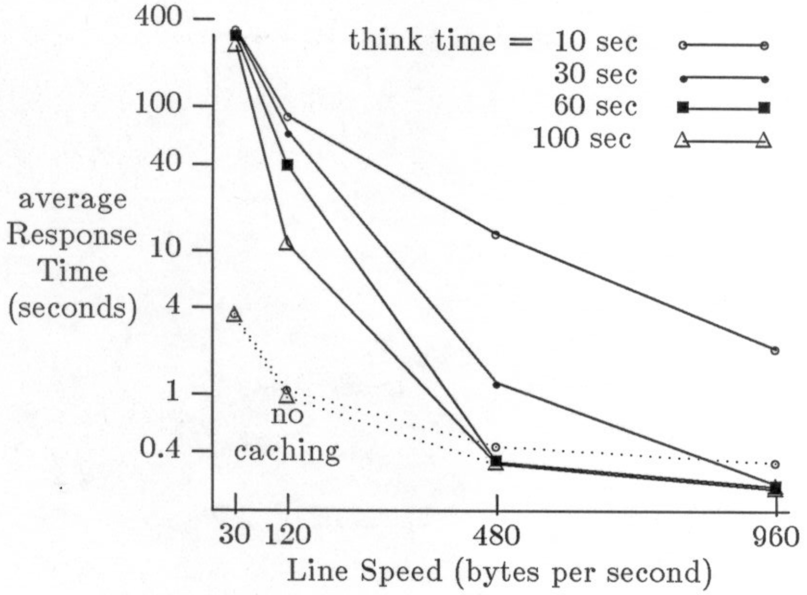

Figure 8: Simple caching compared to standard access for varying think times, uniform reference

With a think time of only 30 seconds, simple caching becomes feasible for 4800-baud users. With 60 seconds, response time is finally reduced to the no-caching level; caching is now free. The improvement at 9600 baud is even more pronounced. We conclude that in an IR environment the performance of caching is dominated by the speed and bandwidth of the communication channel, and the proportion of session time during which that channel is idle.

5.5 Prefetching and local mass storage

Perhaps the most reliable solution to the transmission bottleneck is to cache only while the line is otherwise idle, or at least to place a limit (based on the user's line speed and think time) on the quantity of data which may be cached during any query. This would keep response times reasonable while still permitting data to slowly accumulate in local storage. Another solution is to *prefetch* data. Prefetching may be viewed as caching ahead of time; that is, acquiring data before it is requested at all. This might be done between user queries during a session, or an hour before the user is expected to log on, or further in advance, depending on both the quantity of data in question and its volatility.

We are beginning to study approaches to prefetching. An upper bound on performance can be obtained by supposing that *all* necessary data had been prefetched before the session begins. All queries can then be processed locally. We simulated this with hard disk storage. Average response time is one second, representing an improvement over standard operation for users of 300- and 1200-baud lines, though not for higher speeds (see Figure 3). Even using slower diskettes, response time averages only 1.7 seconds. Because the time required for each transaction is dependent only on the quantity of data involved and the amount of processing, it is predictable and reliable. Users are no longer affected by changing system loads or contention for heavily used files. These results are encouraging, for they imply that a small computer system is sufficiently powerful to run an IR system for a single user.

Several services have begun to distribute the full content of their databases on read-only compact disks (CD ROM) [Chen86]. It is certain that in the near future the use of small, portable, mass-storage media for distributing large quantities of data will increase. This practice is, in a sense, massive prefetching. CD ROM, videodiscs, and optical disks are all potential media for this type of use [Pari83]. While compact disks exhibit slower data transfer speeds than magnetic disks, they still performed well in our simulations, with an average response time of about 3 seconds. Furthermore, a CD can hold 500 to 600 megabytes of data. Having this large store of information available at all times is of enormous benefit to the user. Although the need for remote interactive access to volatile and short-lived data will remain, local mass storage is very attractive for large and relatively fixed bodies of information. When rewritable optical storage becomes available it will be possible to keep a local database current by downloading updates when necessary.

6. FUTURE WORK

More work needs to be done regarding mechanisms for distributing data to users. We expect to investigate policies for prefetching and broadcasting, particularly those which vary dynamically based on observed patterns of usage, and to examine what effect new communication media, such as all-digital phone lines, cable television, and satellite transmission, may have on information system access. We also hope to incorporate promising data distribution strategies into an experimental small computer system accessing a commercial information system. This prototype will likely be implemented on a cluster of Sun workstations which are joined via a local area network. Using a single connection to the information service we can then investigate strategies which take advantage of such clustering. The value of caching and prefetching will certainly increase for clustered users, as each can benefit from data downloaded by others. Decision making, however, becomes more complex. In deciding what data to store, where to store it, and when to discard it, members of the cluster may compete, cooperate via a voting mechanism, or rely on one member which acts as a coordinator.

7. CONCLUSIONS

We have discussed issues concerning the addition of caching capability to workstations accessing information systems. Results of simulations were presented which indicate that while even simple caching strategies can be of benefit when communication lines are not heavily loaded, heuristics restricting the amount of caching must be used to produce acceptable performance at slow transmission rates. The speed of the user (think time) greatly affects the quantity of data that may be cached without seriously degrading performance. It is expected that data prefetching can be used to advantage when transmission is slow, provided that the range of data required is not too broad. Emerging small-scale mass storage media offer the opportunity to bypass telecommunication altogether when accessing long-lived data. All data distribution mechanisms give users greater potential power over the information they acquire.

This research is supported by the State of New Jersey Governor's Commission on Science and Technology, Contract 85-990660-6.

REFERENCES

[Borg83] Borgman, Christine L. End user behavior on an online information retrieval system: a computer monitoring study. In *Proceedings of the Sixth Annual International ACM SIGIR Conference* (Bethesda, Md., June 6-8). *SIGIR Forum* 17, 4 (Summer 1983), pp. 162-176.

[Broo69] Brookes, B. C. Bradford's law and the bibliography of science. *Nature* 224, 5223 (Dec. 6, 1969), pp.953-956.

[Brya81] Bryant, R. M. SIMPAS 5.0 user manual. Computer Sciences Department

Technical Report #456, University of Wisconsin — Madison, November 1981.

[Chen86] Chen, Peter Pin-Shan. The compact disk ROM: how it works. *IEEE Spectrum* 23, 4 (April 1986), pp. 44-49.

[Dunn84] Bill Dunn of Dow Jones: The data merchant [interview]. *Personal Computing* 8, 12 (December 1984), pp. 174-180.

[Giff85] Gifford, David K., Lucassen, John M., and Berlin, Stephen T. The application of digital broadcast communication to large scale information systems. *IEEE Journal on Selected Areas in Communications* SAC-3, 3 (May 1985), pp. 457-467.

[Glos83] Glossbrenner, Alfred. *The Complete Handbook of Personal Computer Communications.* St. Martin's Press, New York, 1983.

[Gold85] Creative Strategies International. Online Data-Base (September 1984). As quoted in Goldmann, Nahum. *Online Research and Retrieval with Microcomputers.* TAB, Blue Ridge Summit, Penn., 1985.

[Jami79] Jamieson, S. H. The economic implementation of experimental retrieval techniques on a very large scale using an intelligent terminal. In *Proceedings of the Second International Conference on Information Storage and Retrieval* (Dallas, Tx., Sept. 27-28). *SIGIR Forum* 14, 2 (1979), pp. 45-51.

[King86] Kingsmill, Brian A. IRS-DIALTECH. In *Downloading Bibliographic Records*, John Foulkes, Ed. Gower Pub. Co., Ltd., Aldershot, England, 1986, pp. 28-33.

[Pari83] Paris, Judith. Basics of videodisc and optical disk technology. *Journal of the American Society for Information Science* 34, 6 (November 1983), pp. 408-413.

[Penn81] Penniman, W. D. Modeling and evaluation of on-line user behavior. Final Report to the National Library of Medicine, Extramural Program Grant No. NLM/EMP (1 R01 LM 03444-01). Dublin, OH, September 1981.

[Rose83] Rose, Denis A. Optical disk for digital storage and retrieval systems. *Journal of the American Society for Information Science* 34, 6 (November 1983), pp. 434-440.

[Smit82] Smith, Alan Jay. Cache memories. *Computing Surveys* 14, 3 (September 1982), pp. 473-530.