A Quasi-distributed Architecture

for

Database Management Systems

Alan L. Tharp
Elizabeth A. Middleton

Computer Science Department
North Carolina State University
Raleigh, North Carolina 27695-8206

Abstract

This paper describes a new architecture for database management systems. This quasi-distributed architecture is a compromise between the traditional centralized architecture and the more recent distributed architecture. As such it provides benefits over both architectures but also has limitations when compared with them. The quasi-distributed architecture is made possible from recent advances in computer technology: it uses economical workstations to place more of the processing at remote sites, and it uses the increased storage capacity of CD-ROMs, WORMs and hard disks to store more data closer to where it will be used. The paper gives an overview of the quasi-distributed architecture and discusses its advantages, disadvantages, and limitations. Although it is not a general purpose architecture, it is suitable for certain situations.

Introduction

"A distributed database is a database that is stored on computers at several sites of a computer network and in which users can access data at any site in the network" [1]. In a centralized database, all data are stored at a single site with access via terminals from local or remote locations. Distributed database management systems have received considerable attention because they have advantages over centralized systems. Providing these many advantages is not, however, without its drawbacks, most of which result from the complexities of implementation when compared with a centralized system. The advantages and disadvantages of a distributed system (which are the opposite for a centralized system) are discussed in the *Comparison* section.

The Quasi-distributed Architecture

The proposed architecture falls between the centralized and distributed architectures, but is closer to the centralized. It is characterized by the distribution of data according to the update frequency. The reason for this data distribution is to reduce the central computer's overhead associated with concurrency control and transmitting data to and from the remote sites. Three data classifications are proposed: R data, which is rarely changed, C data, which is changed constantly, and V data, which is changed periodically, but less than once a day. C data is stored only at the central site. R data is stored at both the central and remote sites, but can only be updated from the central site. V data is stored at both sites, but is updated from the remote site. At the remote sites, the R data, which is high volume, would typically be stored on a CD-ROM, whereas the V data, which is less, would be stored on a hard disk. Economical workstations, from microcomputers on up, would be used at the remote sites to process the data; their arrangement is discussed later in this article. With the cost per MIP being less on a workstation than on a central system, the quasi-distributed architecture has the potential of lower costs when compared with alternative architectures. With much of the retrieval data being placed at the remote sites, the load on the central computer is reduced and therefore a smaller central system is needed. Figure 1 illustrates the basic architecture.

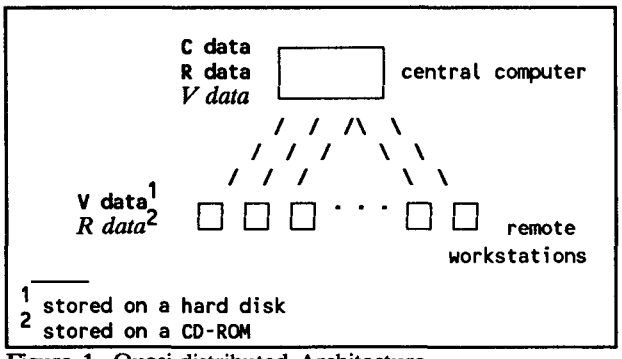

Figure 1 Quasi-distributed Architecture

Applications

A typical application for this architecture is a catalog order business, for example, one which sells a variety of clothing and housewares. The business's catalog is issued quarterly and the prices of items do not change until the next

catalog is issued. The data used are: the catalog (*R data*), prices (*R data or V data*), and the remainder - customer names and addresses, invoice numbers, and shipping labels - (*C data*). The customers may order at remote ordering stations or by calling a toll-free number.

Other applications are flight reservations and auto parts suppliers. The flight reservations application would have flight schedules as *R data*, prices as *V data*, and reservations as *C data*. The auto parts supplier application would have the parts catalog as the *R data* and the prices and suppliers as *V data*. Approved credit customers could also be *V data*. For most applications the *R data* would be stored on CD-ROM and a Bloom filter [2] could be used to determine if changes to the *R data* exist on the hard disk.

Comparisons

Because the quasi-distributed architecture is more centralized than a completely distributed architecture, many of the problems with distributed architectures are eliminated. The advantages of the quasi-distributed architecture as compared with a distributed architecture are:

- Less complex query processing. In a distributed system, because data is stored at several locations in the network, query optimizers must find the locations of the data and optimize the queries to reduce data transfer. In a quasi-distributed system, the location of data is much more structured. If the data is not remote, then it is located at the central computer, optimizing the query is simple.

- Update propagation. In a distributed system with replicated data, propagating updates is complex. In the quasi-distributed system, because most of the data that is updated is in a centralized environment and has only one copy, update propagation is simple. The only problem is making sure V data updates are propagated to all sites and this is easily fixed by broadcasting a message to the workstations that new V data is available.

- Less complex treatment of concurrency. Again, in a quasi-distributed system, the only place where concurrency is a problem is with V data. C data concurrency control is treated just like centralized concurrency control.

- Database design is less complex. In a quasi-distributed system, database design involves mostly finding out which data has a high read frequency and a low add/update frequency for R and V data. In a distributed system, database design is much more complex. Data that a node uses most often should be stored at that node, but another node may use it equally as often. The amount of data replication must be decided.

- Security. Since all users on a quasi-distributed system use the same data, there is no need to secure the data from each other as may be necessary for a distributed system. Network security for the quasi-distributed system is equivalent to network security for a distributed system.

- Recovery. Recovery in a distributed system is made complex by having replicated data. When the system recovers, it must determine which copy of the data is most recent and make all the copies consistent. In the quasi-distributed system, recovery is simple. If a workstation goes down, it just needs to request a new copy of V data from the central computer or another workstation.

- Catalog management. Like a distributed system, the catalog would have to be kept at the remote site, as well as at the central site in a quasi-distributed system.

The distributed architecture, in turn, has advantages over the quasi-distributed architecture:

- Local autonomy. The quasi-distributed architecture does not support local autonomy. It has completely centralized control.

- Easier to increase the size of the system. For a distributed system, this simply means adding another computer to the network. For a centralized system, this means upgrading the central computer. For a quasi-distributed system, increasing the size of the system also means upgrading the central computer, but because some of the data is distributed, the central computer should not have to be upgraded as often.

- Increased system availability. In a distributed system, when a node is down, the rest of the system is still available for data access. In a quasi-distributed system, if the central computer goes down, R and V data can be read, but C data cannot be read and only V data can be updated. The same situation also happens if a LAN gateway workstation is used and it goes down (see the section on *Workstations*).

- Added efficiency. A distributed system has added efficiency because data are stored at the node where it is used most often. This is partially true for a quasi-distributed system, because much of the read-only data is at the remote site and the central computer will not need to be accessed for it. However, accessing data that is updated frequently has the same efficiency as a centralized system.

- Network wide access. A distributed system offers access to data at all locations in the network. A quasi-distributed system does not support access to data outside its own particular network.

- Increased capacity. A distributed system offers increased capacity because processors can be added to the system to make a distributed system more powerful than a single computer. The quasi-distributed architecture offers some increased capacity over centralized system because the workstations have processors.

- Heterogeneous access. The quasi-distributed architecture does not support heterogeneous access.

- Flexibility. A distributed system offers the flexibility that data can be moved to where it is accessed most often. The quasi-distributed architecture does not provide this

flexibility because the data locations are fixed once the frequency of update is determined.

The quasi-distributed architecture has other advantages and limitations when compared with a centralized system.

- Concurrency control for the quasi-distributed architecture is more complex than that for a centralized database because V data is replicated. A record would have to be locked before an update could be done. Then, when the update is finished, it must be propagated to the workstations.

- The system can give the impression of a faster response if the local data is retrieved and displayed while the central data is being retrieved.

Variations

There are several architecture and hardware options that may be feasible with the quasi-distributed architecture:

- there may be an intermediate level of computer hardware between the central computer and workstations. The three levels of hardware may or may not correspond to the three levels of data.

- V data may be eliminated.

- the R data might be stored on a CD-ROM, WORM, or hard disk.

- The V data might be stored on a WORM or hard disk.

- The remote workstations might be connected directly to the central computer or they might be connected to a LAN.

Storage Media

Where should R data be stored? To simplify the discussion of the three alternatives, it is necessary to discuss the difference between the two types of disk rotation that are used, **CAV** (constant angular velocity) and **CLV** (constant linear velocity) rotation. In CLV rotation, data is stored sequentially along a spiral track. The disk's rotational speed changes depending on whether the sector being read is at the center or the outside of the disk so the track can pass the read head at a constant rate. CAV rotation, on the other hand, arranges data in concentric tracks with pie shaped sectors. The disk rotates at a constant speed, so access is faster, but storage space is lost because the outer sectors can only contain as much information as the inner sectors [3].

A CD-ROM is fitting for applications that require storage of large quantities of read-only data at a low cost. The typical storage capacity of a CD-ROM is 540 megabytes. However, CD-ROM access times are slow, around 500 ms [4]. Two reasons for this are that CD-ROM uses CLV rotation, and that the optical read head is heavier than a magnetic read head. The mastering cost for CD-ROM is between $2,900 and $7,300 depending on the database size and turnaround time. The proof disk cost is between $2,500 and $8,000 [5]. Disk replication costs are between $5.00 and $8.50 per disk [3], depending upon the quantity. CD-ROM drive prices range between $500 and $2000 [3]. Because of the high premastering and mastering costs, producing small numbers of CD-ROM disks is not practical.

In deciding whether to use a CD-ROM, it is important to know if the amount of data to be stored is large enough to merit using one, considering the cost of CD-ROM drives and the cost of mastering and distributing the CD-ROM disks.

A WORM is good for applications requiring archival and later reading of data, or applications that need stable storage of large amounts of data but copies are not needed in numbers that merit the mastering of a CD-ROM disk. The storage capacity of a WORM is between 100 and 400 Mb [3]. Access time for a WORM is between that of a CD-ROM and a hard disk, about 150-300 ms [3]. The access time is faster than for a CD-ROM because the WORM uses CAV rotation. A WORM drive costs between $1,500 and $5,000 and a WORM disk costs between $100 and $200 [3]. Many people consider the WORM to be an intermediate technology that will be replaced in the near future by writable optical disks.

A hard disk is suitable for applications requiring fast access and storage of small amounts of data. Access time for a hard disk is typically between 25 and 50 ms. If the amount of R data is too small to warrant mastering a CD-ROM, a hard disk, with a storage capacity of 40 to 120 megabytes, could store it, and there would be the added advantage of much faster access time. The CD-ROM, WORM, and Hard disk are compared in Table 1.

Table 1 Comparison of CD-ROM, WORM and Hard Disk Features.

Feature	CD-ROM	WORM	Hard Disk
Capacity (megabytes)	540	100-400	40-120
Access Time (ms)	400-1000	150-300	25-50
Drive Cost	$400-$2,000	$1,500-$5,000	
Disk Cost	$5 - $8.50	$100 - $200	
Mastering	$5,400 - $15,300 [1]		

[1] the high cost is for a 500 Mb database with 10-day turnaround, and the low cost is for a 100 Mb database with 20-day turnaround.

The CD-ROM would be best for applications that have considerable R data, but if the R data will fit on a hard disk, that would be preferable. If the R data fits easily on the hard disk, no CD-ROM drive needs to be purchased and access time would be faster. The quasi-distributed system can be designed so that if the amount of R data increases, R data could then be placed on a CD-ROM drive. WORMs should only be considered for applications that have a small number of remote workstations because of the time required to copy the database onto the WORM disk and the expense of the WORM drive and disks.

If V data is included in the model, it should be stored on a hard disk. Access time is faster and there would be no cost for the purchase of a WORM drive and disks. However, if the V data does not fit on a hard disk, the WORM drive would be feasible.

Arrangement of Workstations

Direct lines from the workstations to the central computer would improve response time, but the cost of a dedicated line is high; voice grade lines cost from $0.57/mile to $6.10/-mile per month and digital data service costs from $0.72/mile to $15.02/mile per month. (The lower price is for the greater length line.) Also, each dedicated line has a $1,400 to $1,600 one-time cost [6]. A LAN's expense would be only once and, even though a workstation would have to be dedicated to serving the LAN, the LAN would pay for itself after a period of time because it has no monthly expenses as does a direct line. Also, with the decrease in message traffic on the direct line caused by using the quasi-distributed architecture, having only one direct line per remote site is even more practical. Therefore, it would be better to connect the remote workstations with a LAN and use only one direct line to each remote site. The two most prevalent types of LANs are bus and token ring, shown in Figure 2. A bus LAN would be appropriate for the quasi-distributed architecture because that architecture emphasizes fast communication with the central computer rather than communication with other workstations or peripherals. With the token ring, the workstations would spend much time passing tokens around to each other and preference would be given to the workstation next to the gateway because the token would be passed to it every time the gateway received a message. In the example application, small LANs would be needed at the remote ordering stations and large LANs would be needed at the toll-free ordering number.

Conclusions

The quasi-distributed architecture has the advantages of a faster response when retrieving data locally and simple recovery at local sites. It is also potentially less costly than the alternative architectures. It is particularly suited for applications that (1) *have large amounts of data that are read-only* and (2) *small amounts of data that are updated frequently.*

References

1. Pratt, P. J. and J. J. Adamski, Database Systems: Management and Design, Boyd and Fraser, Boston, 1987.

2. Bloom, Burton H., "Space/Time Trade-offs in Hash Coding with Allowable Errors," CACM, Vol. 13, No. 7 (July 1970), pp. 422-426.

3. Buddine, L. and E. Young, The Brady Guide to CD-ROM, Prentice-Hall, New York, 1987.

4. Herther, N. K., "The Silver Disk," Database, October, 1987, p.140.

5. Strukhoff, R., "Made in America," CD-ROM Review, Janunary/February, 1988, p.40.

6. Stewart, Hoyt, IBM, private communication, 1988.

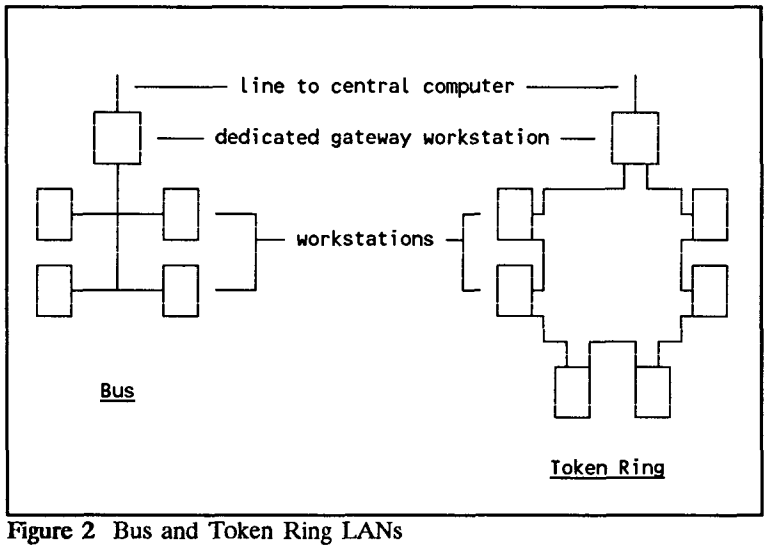

Figure 2 Bus and Token Ring LANs