Modeling Distributed Transaction Response Times As Impacted By In-Storage Buffer Accesses

Tim R. Norton, Sr. Staff Member

Resource Modeling Group
MCI Telecommunications
Tim_R_Norton@mcimail.com

CMG95
Session 352, December 5, 1995

The IBM S/390 Parallel Transaction Server (PTS) has introduced into the mainframe environment many aspects of distributed client/server processing. This paper begins the investigation of one of those aspects; the impact of in-storage buffer misses as additional processors are added to the complex. Additional processors are seen as a solution to both the capacity and availability problems of large transaction workloads. However, as the transactions are dynamically distributed across larger numbers of independent processors, the probability increases that the needed data will not be in an in-storage buffer, thereby increasing the transaction response time due to additional disk accesses. The concepts and principles presented can be applied to any applications moving from a single shared memory computer to multiple systems that do not share memory. This paper outlines the design of a simulation model for application designers to investigate the impacts and trade-offs of utilizing larger numbers of processors.

Background

Processors

In the past, as large mainframe transaction applications have grown, the mainframe processors have also grown at the same or even faster pace. (The term "processor" is generally taken to mean the physical equipment containing individual CPU's (central processing units) that share memory and other supporting functions. IBM uses the term CEC for Central Electronic Complex). Not only have the individual CPU's grown larger, but the number of CPU's within each processor has also increased. However, within the last year the major general business mainframe manufactures, IBM and Amdahl, have stated that they will no longer either increase the size of the CPU's or increase the number of CPU's in a processor. The analysis of the limits within the current mainframe industry is beyond the scope of this paper other than to note that one solution is the IBM S/390 Parallel Transaction Server (PTS) complex and the S/390 Sysplex Architecture (the term Sysplex comes from **sys**tem com**plex**) (IBM 1994). Connecting several processors, either mainframe or PTS, together so that transactions can be dynamically routed and balanced across the processors is referred to as a Parallel Sysplex (IBM 1994, Ricciuti 1994). Even though the PTS processors continue to run the traditional mainframe operating system, MVS, the Parallel Sysplex complex of many PTS's providing transaction processing services begins to look very much like a distributed or client/server environment. These systems are even referred to as servers (Edge 1994, Buzen 1994).

Data Base Systems

The major data base systems that support transaction workloads attempt to keep data base records in buffers in memory to reduce the delays associated with disk accesses. The memory, or in-storage buffers, are accessible by any transaction executing on any of the CPU's within the same processor as the data base system. Because all of the CPU's within a processor have access to the same shared

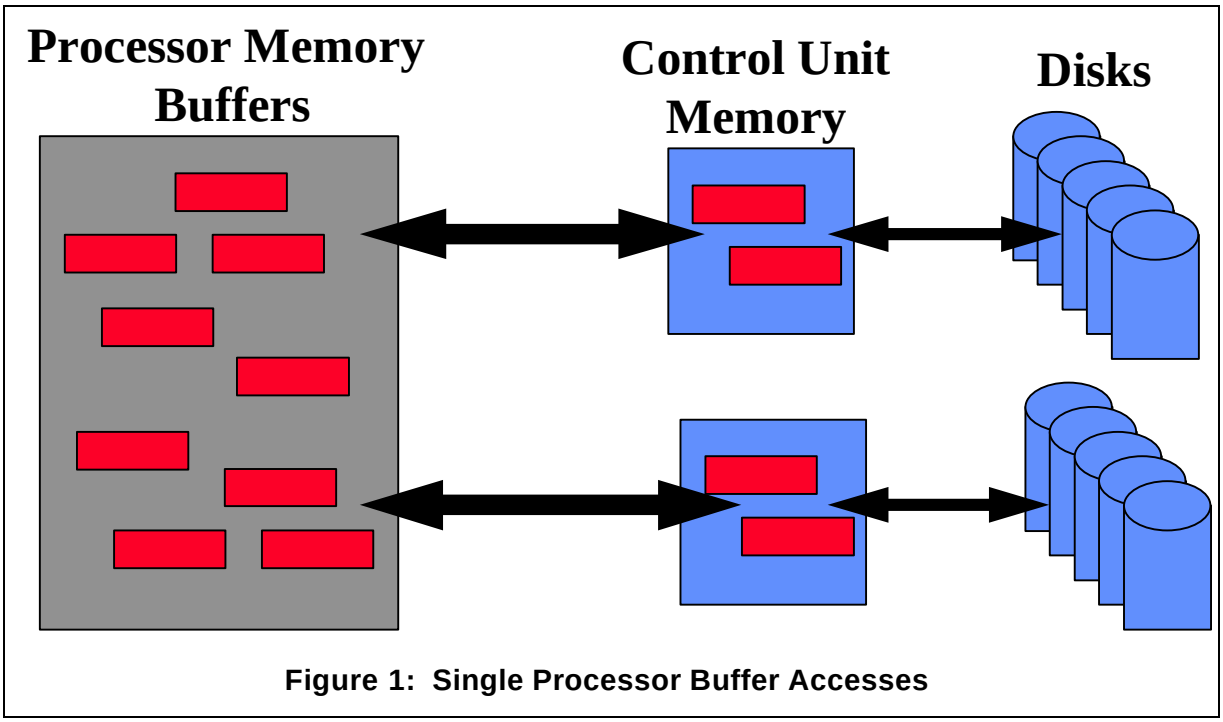

Figure 1: Single Processor Buffer Accesses

memory, which CPU is selected for transaction execution makes no material difference to the transaction.

Coupling Facility

Part of S/390 Parallel Sysplex is a very high speed fiber-optic link (50 MB/sec to 100 MB/sec) between the processors. These links, and the specialized processors that manage the communication across the links, are referred to as the Coupling Facility (CF). The CF will maintain data structures in memory and provide those structures to requesting processors in the Sysplex. The CF was created to allow large numbers of processors to be connected without requiring disk accesses to information such as data base record locks and other operating control structures. This paper is concerned with two aspects of the CF: 1) invalidating in-storage buffers in all other processors whenever the application in one processor re-writes a record, and 2) the future implementations of the CF that will also store the data base records themselves (IBM 1994, Ricciuti 1994).

The Buffer Access Problem

Overview

As data base systems have been optimized to reduce the number of accesses to disk, the transaction level through-put and response times have generally improved. If the number of disk accesses is increased when the transactions are distributed across multiple processors, the benefit of this optimization will be reduced and the transaction response time increased. This becomes very obvious when seen in the form of an example. Assume a transaction requires .005 seconds of CPU time and does 100 data base accesses. Also assume a data base access requires .0001 seconds if the record is found in memory and .030 seconds if it is not; then the total response time can range from 0.015 seconds (.005 + 100 * .0001) to 3.005 seconds (.005 + 100 * .03). Therefore, the higher the percentage of the records that are found in memory, the better the transaction response time. This percentage is referred to as the “buffer hit ratio” and is expressed as the percent of the total records that were found in memory. The term “buffer hit” is used to describe finding the requested record in the in-

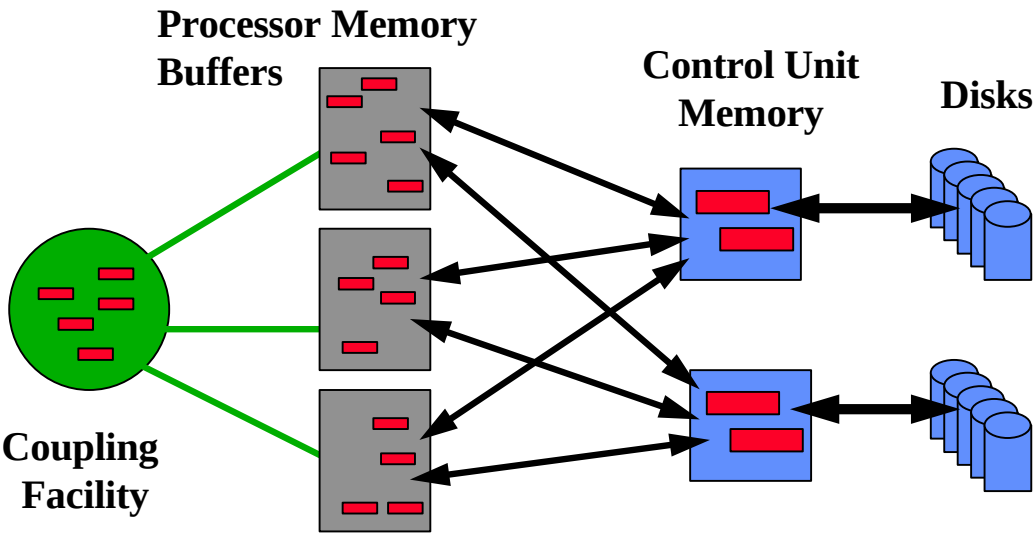

Figure 2: Distributed Processors Buffer Accesses

memory buffer and the term “buffer miss” is used to describe when the record is not found in the in-memory buffer.

Figure 1 is a visual representation of how data records move from the disk devices through the memory (cache) in the disk control units to the buffers in the processor main memory. These buffers are accessible to all transactions executing within that processor.

When transactions that have been executing on a single processor are divided across two or more smaller processors, there will be a decrease in the buffer hit ratio (Buzen 1994) because some percent of the time a transaction that would have used a record just loaded into a buffer by another transaction will be dynamically routed to a different processor and will have to re-load that record from disk. Future releases of the Coupling Facility and data base systems enhancements will reduce this penalty by re-loading the record over the very fast CF links, but it will never be as fast as an in-memory hit. Access across the CF link appears to have a response time somewhat faster but of about the same magnitude as a control unit cache hit. “Performance of the JES2 checkpoint on the coupling facility is equivalent to the performance of the checkpoint on the DASD.” (IBM 1995). There has been some discussion in the literature about the impacts of dynamically routing the data base requests, but authors either focus on the functional impacts and the individual server performance (Ferguson 1994) or note that “This [reduced buffer hits] can have an adverse effect on performance, but the effect may be small in practice.” (Buzen 1994) Much of the analysis has been directed toward determining the impact of the smaller CPU’s (Buzen 1994). However, as shown be the example above, even when most data base accesses are satisfied from buffers in memory, the CPU time is a small part of the total transaction response time. When more of the data base accesses are not satisfied from buffers in memory, the CPU time becomes almost unnoticeable. In the above example, reducing the size of the CPU by a factor of four will have a smaller effect than reducing the number of buffer hits by one access (.005 * 4 = .020 < .030).

Figure 2 is a visual representation of how data records move from the disk devices through the disk control unit cache to the buffers in the main memory of the distributed processors. Notice how the main memory of the distributed processors has fewer records than with the single processor in Figure 1. The Coupling Facility represented on the left side of Figure 2 also shows records (the small rectangles) to represent the future feature where the CF will act like another level of cache between the CU and main memory. Routing transactions to specific processors based on transaction content can be

effective in reducing the impact of fewer buffer hits, but may have serious implications in terms of processor management and utilization; discussion of either being beyond the scope of this paper which assumes an even (random) transaction distribution across all of the processors in the complex.

One other study has shown a direct correlation between transaction response times and the number of processors. As the number of processors goes up, so does the response times. The response time on eight processors was more than twice that of a single processor for the same workload (Dan 1994).

Details

What can happen

Table 1 shows what can happen whenever an application makes a database access and the approximate time required to move the data into the application's work area. All of the times are approximate. Because the data cache feature has not been implemented, the CF hit time is a guess by the author based on control unit cache access times. If the record is not found in the buffer, the next location in which it can be found is the CF (when implemented); after that the cache in the disk control unit; then the access must go to the disk. It is important to note that the absolute times are not as important as the relative order of magnitude between them.

If the Coupling Facility support is not available, then device utilization will go up as the buffer hits go down, which will increase the demand for the control unit level cache; which in turn increases the device response time.

What Causes Buffer Misses For Already Read Records

If a record has already been read into the buffer on one processor, there are several reasons that would cause a buffer miss. The response times for that access will depend on where the record is and the related access times. Hundreds of in-memory accesses can be done faster than a single access that must go all the way to the disk. The major causes of a buffer miss for a record that has been recently read into a memory buffer are:

1. The transaction is dynamically routed to a different processor where that record has not been accessed before.

2. The record is updated in a different processor and the current copy must be invalidated, discarded (the invalidate overhead time must also be added to the access response time) and a new copy obtained.

3. The record is not accessed often enough and migrated out of the buffer (this is a function of the processor memory size).

Buffer-Miss Drivers

The drivers are the characteristics of the application and of the system that control how well or how poorly the application utilizes the in-memory buffers, and therefore, how the application performs. The major drivers are:

- Locality of reference
- Reuse of records
- Number of physical processors or CEC's
- Read-to-write ratio
- Memory size

Locality of reference of records is the description of how close together the records are that are being used within a short time-frame. The closer together the referenced records are, the more effective the buffer read-ahead will be, either in the control unit or the data base system. Locality of reference will decrease from the processor point of view as additional processors are introduced into the complex because the transactions most likely to take advantage of the read-ahead will distributed across an increasing number of other processors.

Reuse of records is the description of how often a given record is referenced. The greater the reuse of records, the more likely a transaction will find the needed record already in memory. At the extreme, if every transaction read the same record, the impact of additional processors would be small because once that record is read into memory, no additional disk accesses would be incurred. On the other hand, if no record is ever reused and the locality of reference is very poor, buffering would be essentially useless and all accesses would go all the way to the disk device. Where between these two extremes does an application fall?

The number of processors, or CEC's, will determine the probability that a transaction will be routed to a processor where the required record is already in memory. Adding processors will impact different transaction workloads differently depending upon whether the workload tends to stabilize over time and thereby improve the in-memory buffer hit ratio after some ramp-up time. If the workload does not stabilize, then the record reuse rate will never approach that of the single processor environment.

The record read-to-write ratio is the number of times that record is written divided by the total number of accesses (reads plus writes). Every time the record is written, the CF must notify all of the other processors that their copy of that record is no longer valid and must be discarded. Even if two sequential transactions read the same record on one processor, a transaction executing on a different processor and updating that record can cause the second transaction to access the disk for the record. The CF will at some point in the future address this problem by passing the updated record to the other processors, but the current implementation does not. Even when it does, the CF access times will be greater than the in-memory buffer access times.

The processor memory size will determine the number of records held in the buffers. Transactions reading the same record, but on different processors, will require up to x times additional memory for that record (where x is the number of processors minus one) to hold duplicate copies. The arrival rate of transactions, along with the other workloads supported by the same processor complex, will determine how likely it will be to find the record in memory. A lower arrival rate of the application of interest combined with higher activity of other workloads will cause a decrease in buffer efficiency from the point of view of the application of interest.

The response times can more than double when moving from one processor to eight processors. If the buffer size on each processor is not large enough to hold all of the frequently referenced records, the buffer hit probability will suffer and response times will rise (Dan 1994). Maintaining this level of memory could require each processor to have an amount of memory

TYPE OF ACCESS	RESULTING ACTION	RESPONSE (SECONDS)
Buffer hit	None	less than .001
Buffer miss	Coupling Facility access	.001 to .003
Coupling Facility miss	Control Unit cache access	.003 to .005
Control Unit miss	Disk access	.020 to .030

Table 1: Types of Data Base Accesses

equal to that of a single processor; eight times the original memory. Unfortunately, even this additional memory does not reduce the response times to the level of a single system due to other factors such as aborted transactions and buffer invalidations (Dan 1994).

Not Considered

There are many other areas dealing with operation system control structures, such as data base lock delays, file enqueue delays and security system accesses that will be impacted by distributing the transactions across multiple processors. Although very important, these areas are outside the scope of this paper.

Modeling the Problem

Overview

From the discussion in the sections "What Causes Buffer Misses For Already Read Records" and "Buffer-Miss Drivers", the reader should begin to appreciate the complexity of understanding the overall impact of changing just one of the drivers. When a application is moved from a single processor environment to a multiprocessor environment most, if not all, of the drivers will change making that overall understanding almost impossible. The model described in "Designing the Model" is designed to take the changes of all the drivers into account when recalculating the new transaction response time. In addition, the model will simulate how the transaction workload interacts with the physical systems and collect information that would otherwise not be available until sometime well into the testing phase. By collecting this information and predicting the impacts earlier in the development cycle, all participants will make better design and implementation decisions.

Analytic modeling tools are used to describe the changes to the average transaction response times (Buzen 1994). In analyzing this problem, the major drawback to using a analytic model is the inability to deal with the individual transactions. Predicting the changes to the buffer hit ratios and record reuse rates requires that the model understand the order in which the transactions are executed.

A simulation model describes individual transaction behavior, the impact of the order of transaction execution and outliers; the small number of transactions whose response time is so different from the majority as to effect the average but with such a small volume as to not impact other areas.

Modeling Tool

Although there are many simulation tools to choose from, the author selected the SES Workbench from SES Inc. in Austin, TX, because it provided most of the required features and it had already been purchased by another department and was available to the author at no additional cost. Other simulation tools are available and could be used.

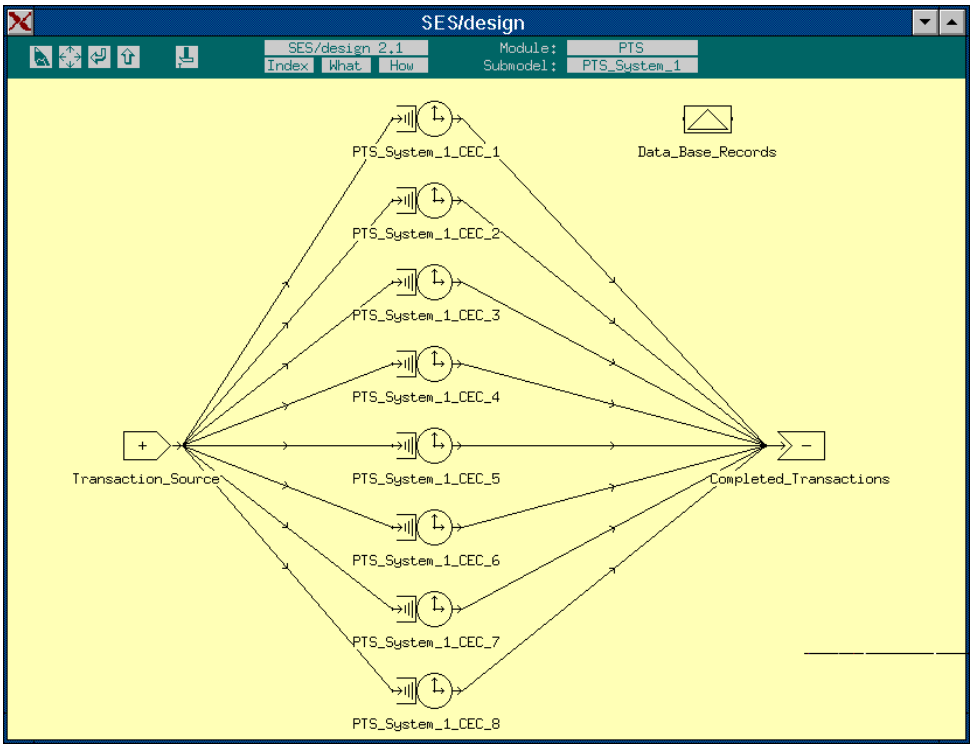

Figure 3: SES PTS Workbench

Designing the Model

The model must allow the user to change the dominate aspects of the simulation from one execution to another. The major areas that require this control are: number of processors, buffer locality of reference, processor memory size, and transaction profiles.

Variable Number of Processors

The initial version of the model will assume a random distribution of transactions across all of the processors. The model will use the number of processors based on an input parameter and will distribute the transactions evenly across that number. Future implementations could allow the distribution to be skewed by transaction content, but this would complicate the model to such a point that it would no longer be possible for the model to be completely parameter driven. Figure 3 shows the high level model with each of the processors in the center. The source symbol on the left side (TRANSACTION_SOURCE) controls the transaction arrival rate and the connecting lines define the probability a transaction will be sent to that processor (PTS_SYSTEM_1_CEC_1-8). The sink symbol on the right (COMPLETED_TRANSACTIONS) exists simply to remove the transaction from the simulation.

Control Over Buffer Locality of Reference

The model must allow the user to control the locality of reference and record reuse, either by detailed transaction workload profiles or by the selection of different probabilities. Although not as accurate, using probabilities allows selecting worst case (a transaction doing random accesses) and best case (a transaction doing accesses very close together and to the same small number of records). The level of accuracy of the model in this area is extremely dependent on the accuracy of the information describing the way the buffers are used. The best information will be collected from an existing application currently executing on a single processor and will include data for each transaction with an exact record identifier. This type of information, while

very desirable, is very costly to collect and may require application changes to get the required level of detail. The accuracy of any other method will be dependent on how well the generalization of buffer accesses matches the true behavior of the application.

Processor Memory Size

The model must allow the amount of memory per processor to be varied as well as using different memory management techniques. As the number of processors increases, the total installation memory investment will increase dramatically to provide the same level of buffer hits because of the large number of duplicated records across all of the processors. The higher the current single processor buffer hit ratio, the greater the total memory required. On the other hand, if the current application's buffer hit ratio is very low, then the existing memory size can be divided across the new processors, each getting an equal share with only a small increase for the more active records. Thus a poorly performing application will continue to perform poorly.

Transaction Profiles

Describing the profile of an existing transaction workload can be both interesting and challenging. The model must allow the user to vary the amount of CPU time, the number of disk accesses per transaction, the transaction arrival rate and the impact of any other interference workload.

Using the Model

What Will It Do?

The model will provide predictions about changes in the application response times and buffer hit ratios in response to changes the number of processors and the size of the processor memory.

Response Time Predictions

The model will predict the response time range of different transaction workload mixes as the number and configuration of resources changes. The user can set the number of processors to one to allow for calibration against the existing workload. Then, the different resources can be changed to observe the impact on transaction response times.

Buffer Hit Ratio Predictions

The model will predict, for a given transaction workload, how the buffer hit ratio changes as a function of the number of processors and memory size. One goal would be to maintain the same or greater buffer hit ratio while adding a large number of additional processors, based on the assumption that changes to the buffer hit ratio are a general predictor to changes in the transaction response times.

The model can be used to determine the different memory sizes based on the buffer hit ratios and transaction response times. The user should be able to continue to increase the memory until there is no significant change in either the hit ratio or response time. This would identify the optimal memory size for that transaction workload.

What are the Benefits?

The benefits to modeling the behavior of an existing application before committing to moving that it are substantial. Some applications may move very easily to a parallel distributed processor environment while other applications may have such unusual data access patterns that such an environment would be cost prohibitive. An application with a high percentage of write updates randomly across the data base might fall into the latter category.

Additional Processors Analysis

Predictions of response times for different numbers of processors would provide management cost justification information showing the increased benefit at each level. The model will not address such issues as availability and processor failure, but by identifying the number of processors to support the application, management will be able to add in the cost for those additional functions.

Data Base Re-design Analysis

Although not identified empirically, the model can assist in identifying data base design issues and the value of re-design to improve the

buffer locality of reference and record re-use. For example, if no reasonable amount of additional memory can bring the buffer hit ratio up to the level seen in the single processor environment, then the application may be designed such that groups of transaction need to execute on the same processor. One solution could be transaction routing based on content rather than to balance the load across all of the processors, but this would also have an impact on the total processor utilization.

Additional Memory Analysis

Is lots of memory in small cheap packages better or worse than the original single large expensive package? Although distributed processor memory may be substantially less expensive than mainframe memory, it still has a significant cost in very large quantities. The ability to determine during the planning stages what the cost of the major application performance drivers will be should enable management to make a better case for moving the application and should enable the implementers to move the application more effectively.

Conclusion

There has been very little published discussion about the impact to transaction response times when moving applications from a single shared memory processor to some number of distributed processors that do not share memory. The Buffer Access Problem, as described in this paper, identifies what may be a significant bottleneck to moving applications. Modeling the problem provides a large amount of information about the behavior of the application in different environments without the investment and risk of actual implementation. The major areas of discussion of the IBM S/390 PTS have centered around the effects of moving the application from a few large CPU's to many smaller ones. The author believes this is due to the higher level of visibility that processors receive (because of cost) and because the processor component is much easier to isolate from the other aspects of the application workload. The impact of the Buffer Access Problem has the potential to far overshadow the processor performance issues. As processor manufactures continue promoting the benefits of distributed processor CPU's, the Buffer Access Problem will become an increasingly larger reason for transaction response time delays for distributed workloads.

Selected Bibliography

Buzen, J. P., and Shum, A. W. 1994. Best/1 Considerations for IBM S/390 Parallel Transaction Servers. BGS Systems Technical Report, 94148-01

Dan, Asit, Daniel M. Dias, Philip S. Yu. Buffer Analysis for a Data Sharing Environment with Skewed Data Access. *IEEE Transactions on Knowledge and Data Engineering*, Vol: 6 Iss: 2 p. 331-337. April 1994.

Edge: Work-Group Computing Report 1994. Large scale computing: IBM continues to drive transformation of large scale computing; new IBM S/390 Parallel Enterprise Servers. 5(226): 5-6.

Ferguson, Mike. 1994. Breaking up is hard to do. Database Programming & Design 7(11): 33-40.

IBM. 1994. S/390 Parallel Sysplex Presentation Guide, HTPG Package, Large Scale Computing Division.

IBM. 1995. S/390 Parallel Sysplex Performance, GG24-4356-00.

Ricciuti, Mike. 1994. How the parallel mainframe works. Datamation 40(11): 71-72.