LBL-12412

CONF-810467--1

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

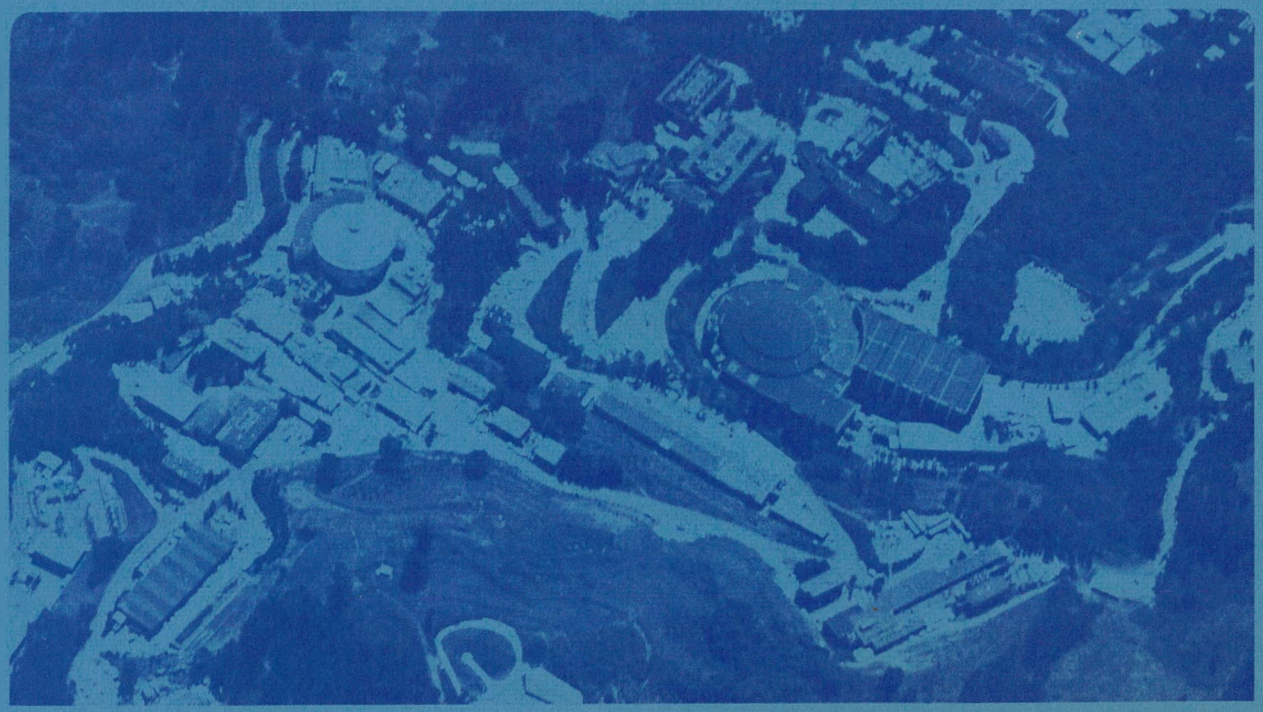
Physics, Computer Science & Mathematics Division

To be presented at the SIGMODE International Conference on Management of Data, Ann Arbor, MI, April 29-30, 1981

THE EFFECT OF TARGET APPLICATIONS ON THE DESIGN OF DATABASE MACHINES

Paula Hawthorn

March 1981



Prepared for the U.S. Department of Energy under Contract W-7405-ENG-48

The Effect of Target Applications on the Design of Database Machines

by
Paula Hawthorn
Computer Science and Mathematics Department
Lawrence Berkeley Laboratory
University of California, Berkeley, CA 94720

ABSTRACT

Specialized, single function processors can be built to be faster and cheaper than general purpose processors. Most database machines use such special purpose processors to manipulate data, with a general purpose managing processor to control the special purpose processors and perform utility functions. In this paper, the organization and use of these data manipulation processors is explored. Database machines are classified into single data manipulation processor systems, multiple disk-associated data manipulation processor systems, and multiple cache-associated processor systems. Examples of actual database machines are given for each category.

Application types are classified into business, bibliographic search, and statistical analysis systems. A metric is developed to compare the performance of the categories of database machines with respect to the application types. The metric is the effective instruction rate, which is comparable to the instruction rate (millions of instructions per second) for conventional computers. It is shown that the effective instruction rate is highly sensitive to the proportion of work performed in the database machine's data manipulation processors. Therefore, determining the work that must be performed in the machine's managing processor is found to be important to determining the performance of the machine.

Database machine performance for each category of database machines is compared for each type of application. It is shown that single processor systems are best for business applications; that disk-based multiple processor systems are best for bibliographic search applications; and that hybrid systems are best for statistical analysis applications. Therefore, the design of the organization of data manipulation processors is shown to be application-dependent.

1. Introduction

A database machine is a computing system especially designed for data management. Several such machines have

been designed and/or implemented. Examples of database machines are: RAP (Relational Associative Processor), [SCHU78]; DBC (Data Base Computer), [BAN78A]; DIRECT [BORA79B], CAFS (Content Addressable File Store), [BABB79, COUL72], CASSM (Context Addressed Segment Sequential Memory) [LIPO78], VERSO [BANC80], and IDM (Intelligent Database Machine), [BRIT80, EPST80]. In the discussions of designs of database machines there have been few attempts to compare the effect on performance of the fundamental design differences of the machines. [ROSE77], [SU79], and [SMIT79] are descriptive comparisons of several machines without performance estimates. In [HAWT80] several database machines were compared, and total query times for each machine were estimated on the basis of benchmark sets of queries. However, the comparison was on a per-machine basis.

In this study an individual machine comparison is not attempted; instead, we explore one fundamental database machine design parameter (the use of special processors for data manipulation) and show that it is inherently application dependent. If all the possible database machine differences are considered, the design space is multidimensional and each machine is a point in that space. In isolating a single design parameter and exploring the performance consequences of variations of this parameter, we study a particular dimension of the design space.

In Section 2 three machine categories are proposed: single data manipulation processor (DP) systems, multiple

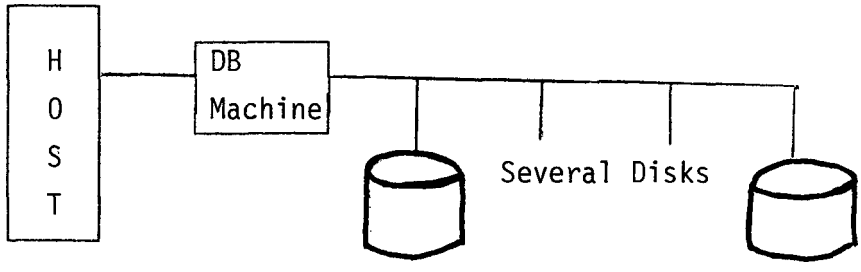
disk-associated DP systems, multiple cache-associated DP systems, and hybrid systems. A metric is developed, in Section 3, to compare machine performance with respect to the use of data manipulation processors. The metric is the effective instruction rate, which is equivalent to comparing instruction rates (millions of instructions per second) for conventional computers. Section 4 contains a comparison of database machine categories with respect to three generic application types. These are: business, bibliographic and statistical analysis applications. It is shown that the best database machine for each application has a different design. The best machine for business applications is shown to be the single DP system; the bibliographic applications are shown to run best on a disk-associated DP system; the cache-associated and hybrid systems are found to be best for statistical analysis applications. Section 5 concludes that there is no single best database machine design. Instead, it is shown that the design of the data manipulation processor system is application dependent.

2. Design Differences of Database Machines

Database machines are ordinarily back-end systems, where the front-end (or "host") directly communicates with users and off-loads to the database machine all data management functions. Most database machines have the general form shown in Figure 1. They include one or more processors, a memory cache, disk controllers, and disks. The host

sends transactions to the database machine which executes those transactions and sends any data requested back to the host.

Figure 1: Database Machines



There are several advantages gained by the use of a database machine. The first is portability. Because the host-resident program to connect the database machine to the host can be simpler than a DBMS, a database machine can be easily connected to many types of hosts and operating systems. This is significantly different from software-only data management systems which either must undergo extensive changes as they are implemented on different systems, or must sacrifice efficiency for portability.

The second advantage of database machines is that data can be more secure, since the only access to the data is through the machine. Another advantage is that the machine can be simultaneously connected to many hosts, thus providing a central database facility.

The fourth, and perhaps most important, advantage for database machines is the low cost, high performance gained by the use of dedicated systems and custom built hardware. The data management system in a database machine does not contend with any other processes for control of resources. Therefore such operating system functions as buffer management, process management, disk space allocation, disk read-ahead, and file management are totally under the control of the data management system and can be specialized to conform to the data management system's needs.

Specializing the operating system functions leads to performance gains [HAWT79], but even better performance is possible with the use of especially designed, custom-built

processors.

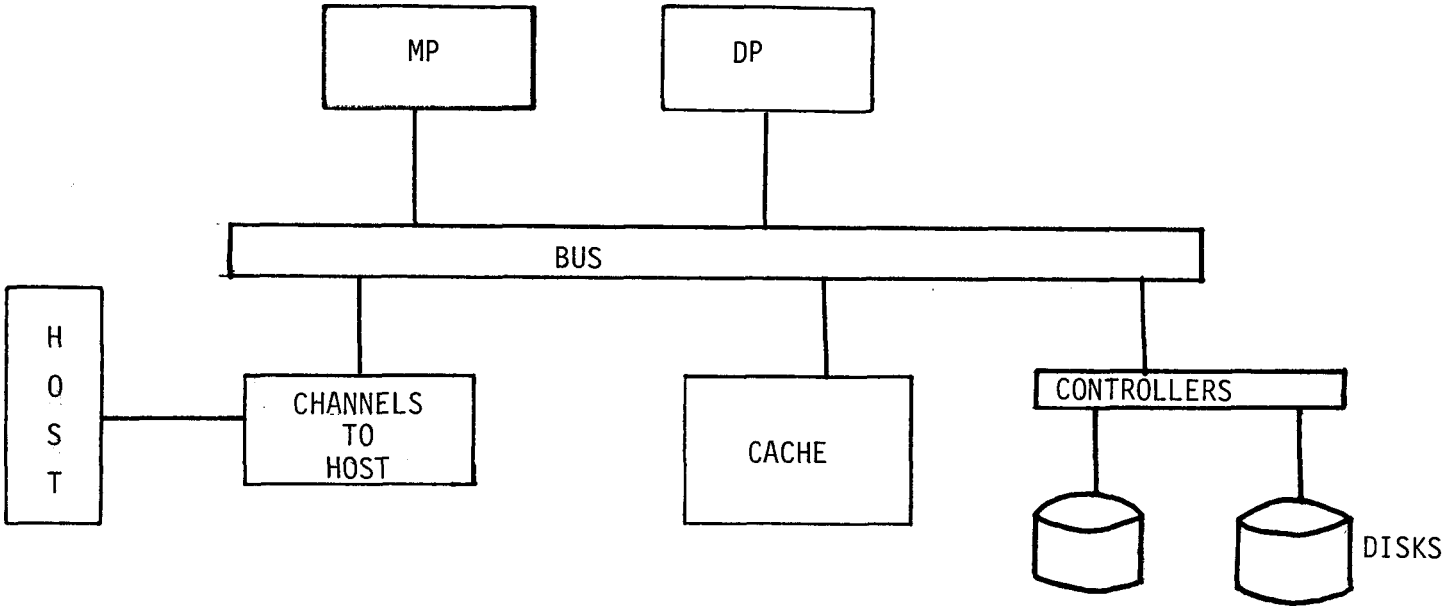
A special-purpose logic device, one that must perform only a few functions, can be built to obtain higher performance at lower cost than a general-purpose device [SU75, BABB79]. Many of the functions of data management systems are simple, repetitive, and easily built into low cost, high speed hardware. Examples of these functions are: compare for equal, less than or greater than a given value; simple arithmetic functions; simple compression techniques. Most database machine designs use one or more especially designed data manipulation processors in order to improve performance. In the IDM the data manipulation processor is called a "database accelerator" [BRIT80]; in RAP it is a "cell processor" [SADO78]; in DBC it is a "mass memory processor" [BAN78B]; and in DIRECT it is a "query processor" [DEWI79]. In each system the high performances claimed are based on the use of the data manipulation processors [EPST80B, BAN78A, OZKA75]. However, in [HAWT80] it was pointed out that the performances of the database machines are dependent on the query type. Therefore, we will examine the fundamental issue of the use of the data manipulation processors to determine if the optimal organization of these processors is dependent on the application type.

2.1. Single Data Manipulation Processor

In a single data manipulation processor (DP) system the one DP is used in conjunction with a slower, general purpose

managing processor. The Intelligent Database Machine (IDM) [EPST80], a product of Britton-Lee, Inc., is an example of a single DP system. Figure 2 shows the architecture of the IDM.

Figure 2: IDM



The IDM is designed around a high-speed bus. There are channel processors that communicate with the host; controllers that communicate with the archival store (disks); a RAM cache; one 10 mips DP; and a slower managing processor (MP) [BRIT80]. The MP coordinates the processing of commands and performs the functions not provided in the DP. The DP processes data as it streams from any of the disks, or it can process data that resides in the cache. In the IDM the controller, channel, and managing processors can all be executing concurrently with the DP. However, the single DP design forces the limiting case to be essentially SISD (single instruction single data stream) since there is only one DP to act upon the data.

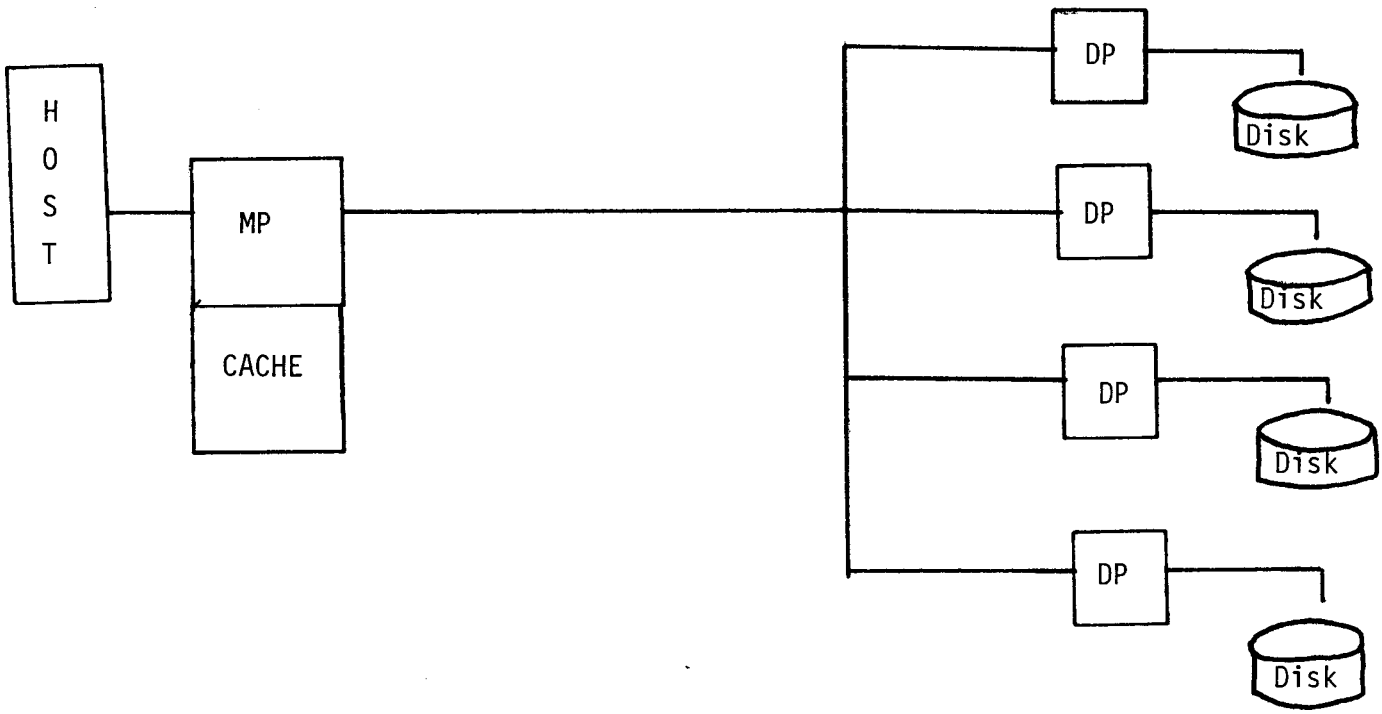
2.2. Disk-associated Multiple Data Manipulation Processors

There are two approaches that have been taken with multiple DP systems. The first is to partition the archival store into separate "data cells" and to associate a DP with each cell; the second is to partition the cache, and to associate a DP with each cache cell. The former we call disk-associated DP systems; the latter, cache-associated DP systems.

Figure 3 shows a disk-associated multiple DP system. The data is processed as it streams from the archival store; only the processed data is sent to the managing processor. For example, if the command is a restricted projection, only

the desired attributes of the qualified tuples will be seen by the MP. The Content Addressable File Store (CAFS), a product of ICL [BABB79] is a noteable example of a disk-associated DP system. The partition size for CAFS is a single track of a moving-head disk, but the size of the partition can clearly be a cylinder or even an entire set of disks.

Figure 3: Disk-Associated DP's



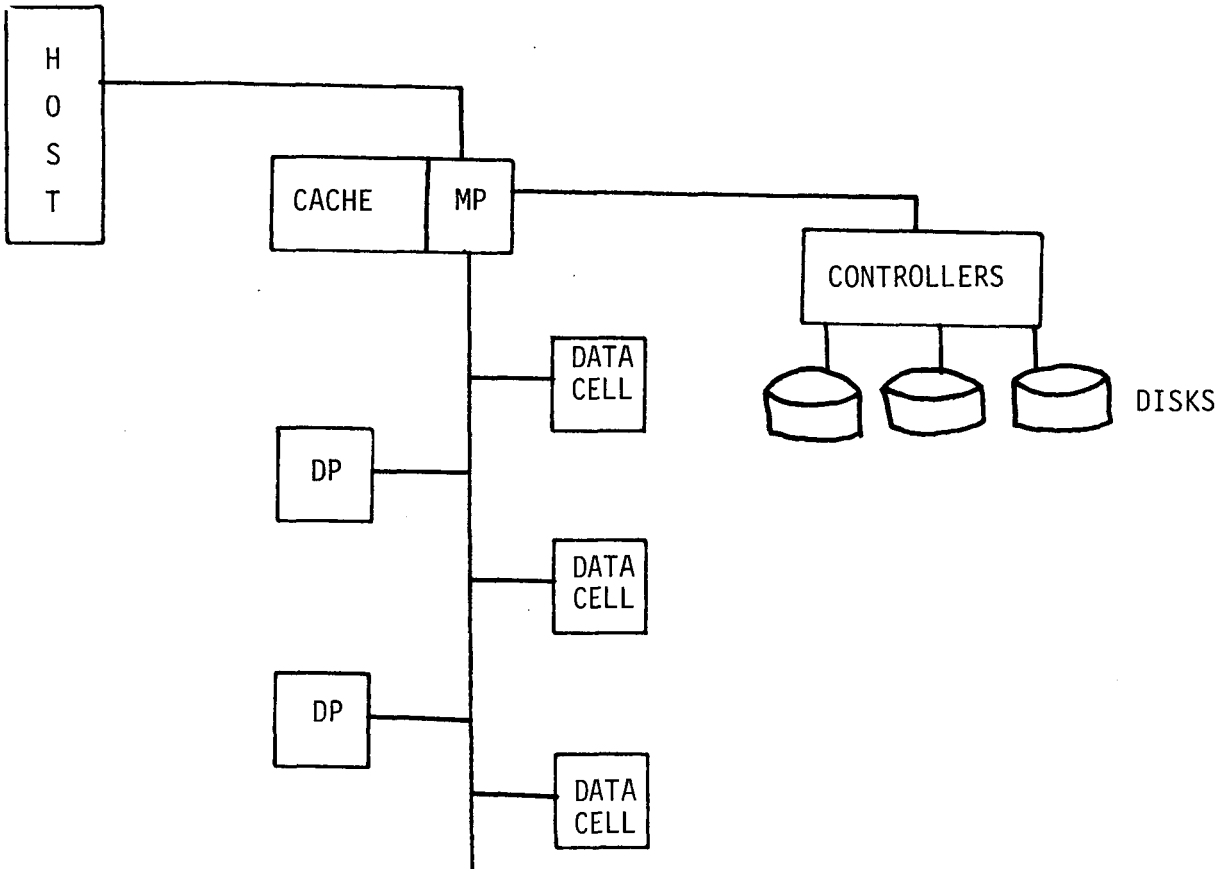
Both DBC and VERSO also employ disk-associated DPs.

Note that Figure 3 shows a multiple processor system with centralized control. Since the controlled processors (DPs) are much faster than the controlling processor (a general purpose processor, the MP), the control functions must be simple to insure that the system does not bottleneck at the controlling processor. This is done in CAFS by making the system a single instruction multiple data stream (SIMD) system: all the DPs execute the same query.

2.3. Cache Associated Multiple Data Manipulation Processors

A cache-associated multiple DP system is shown in Figure 4.

Figure 4: Cache Associated DP's



A typical cache-associated system is DIRECT, which was developed at the University of Wisconsin. In DIRECT, the cache is split into n partitions; there are n/2 processors associated with the cache, and a single MP. More partitions than processors allows partitions to be used as data buffers. The MP matches a DP with a cache partition through the use of a modified cross-point switch. DIRECT is a multiple instruction multiple data stream (MIMD) system: the DPs can all be operating on separate queries, or all on the same query. DIRECT is known to bottleneck at the managing processor [BORA79]. RAP is similar to DIRECT, but has a SIMD architecture and is less likely to bottleneck at the MP.

2.4. Hybrid Systems

It is possible to design a database machine that includes both disk-associated and cache-associated DPs. The Data Base Computer (DBC), developed at Ohio State, is a limited example of a hybrid computer. It contains disk-associated DPs (in the mass memory processors) and limited join support cache-associated DPs (in the Post Processing Unit). It also contains special processors for processing transaction overhead.

3. Effective Instruction Rate

The metric used to compare the performances of the different architectures is the effective instruction rate. This metric is chosen because it gives insight into the

relative performances of various designs without the complexity of such questions as the arrival time for transactions, the exact amount of work per transaction, and so on. The calculation used for effective instruction rate is the following:

(EQ1):

$$ER = 1/\{p/(k * f) + [(1 - p)/s]\}$$

where

ER = effective instruction rate
p = the percentage of the total instructions performed in the data manipulation processors
k = number of active data manipulation processors
f = speed of data manipulation processors
s = speed of the managing processor

EQ1 is derived as follows:

The percent of instructions performed in the DPs (p) divided by the aggregate instruction rate of the DPs (k active DPs times f mips per DP) gives the number of seconds per instruction for the DP portion of the transaction. A DP is an "active DP" if it is participating in the query processing. This is an important distinction because DPs may be associated with physical partitions of data. If the data for a particular query resides in only a few partitions, only a few of the DPs will be "active" for that query. The number of seconds per instruction for the MP portion of the command is calculated in a similar fashion. Then the total is inverted to obtain the total effective instructions per

second, ER.

To illustrate the above, assume that there are 4 active DPs with an instruction rate of 10 mips, an MP with an instruction rate of 1 mips, that an average transaction is 100 million instructions, and that on the average 20% of each transaction is performed in the MP. Then:

```
number of instructions in DPs =
        .8 * 100M = 80M
time for DP part of command =
        80M / (10 * 4 mips) = 2 seconds

number of instructions in MP: 20 M
time for MP part of command =
        20M / (1 mips) = 20 seconds

total time to perform query: 22 seconds
effective instruction rate:
        100 M / 22 = 4.5 mips
```

The above calculation assumes that the time in the MP is not overlapped with the time in the DP because it is not clear how much overlapping can be performed. Since the MP time so overwhelms the DP time overlapping does not appear to be significant.

Figure 5 shows the graphs of EQ1 for three values of k, the number of active DPs (k = 1, k = 16, and k = 96) when the instruction rate of the MP is 1 mips, and of the DPs is 10 mips. The figures of 1 mips and 10 mips were chosen because they appear to be representative of the best times that can be obtained for, respectively, a small, general-purpose processor, and a special purpose data manipulation processor. The figures of k = 1, 16, and 96 were picked to show boundary conditions. We note from Figure 5 that the

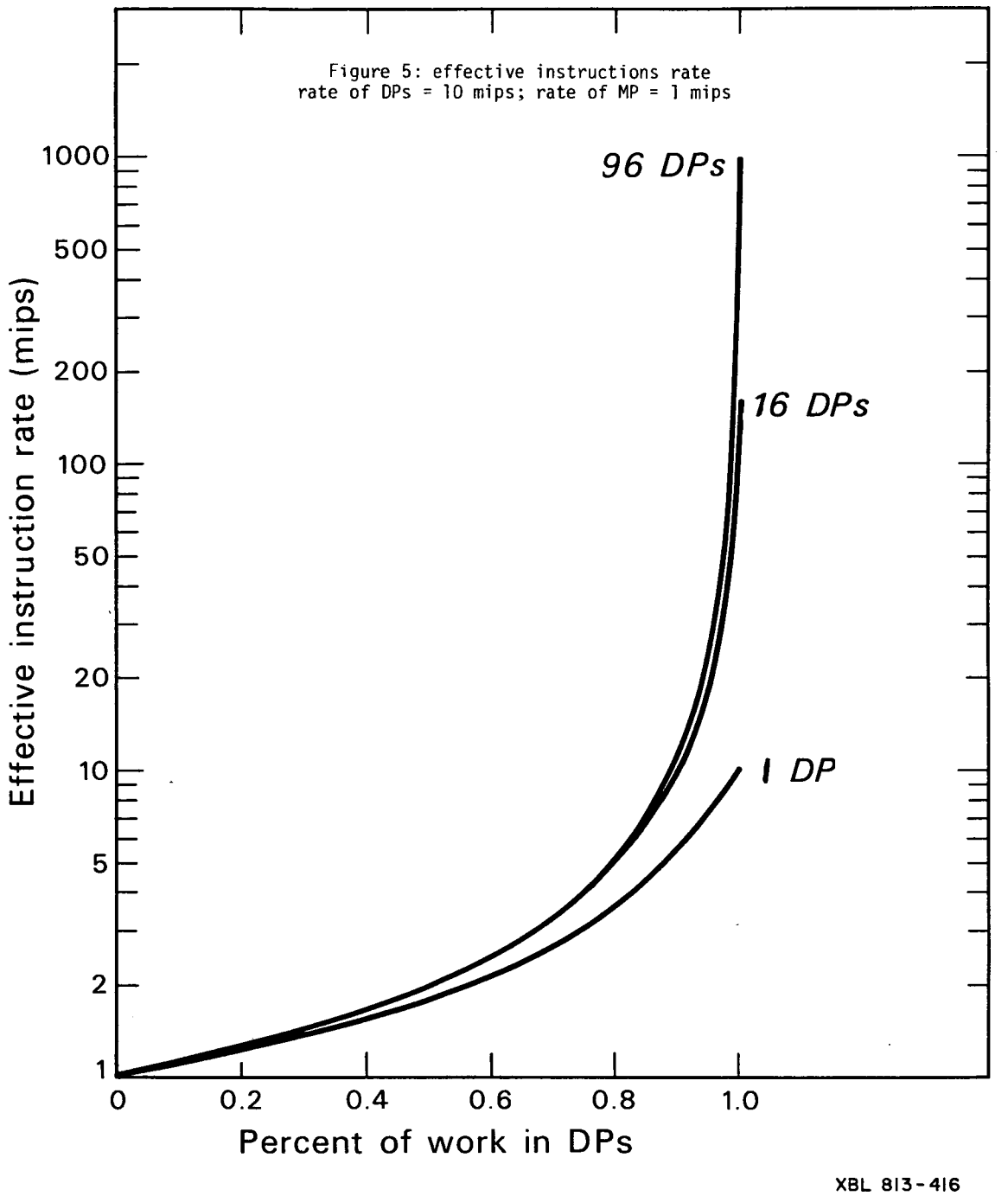


Figure 5: effective instructions rate
rate of DPs = 10 mips; rate of MP = 1 mips
96 DPs
16 DPs
1 DP
Effective instruction rate (mips)
1000
500
200
100
50
20
10
5
2
1
0
0.2
0.4
0.6
0.8
1.0
Percent of work in DPs
XBL 813-416

effective instruction rate is heavily affected by the amount of processing in the MP.

The curves in Figure 5 are surprising. We would expect that the addition of extra processors to a database machine would have a nearly linear effect on the performance of the machine; that is, if we double the amount of data manipulation processors, the performance of the machine should double. This seems especially obvious because EQl does not account for the increased management functions due to adding processors. But we note from Figure 5 that the effect of adding more processors is seen only when the per cent of work in the DPs is extremely high. However, if the overhead is high, systems with additional DPs achieve only a slight increases in the effective instruction rate because the dominant factor becomes the overhead (the amount of work performed in the MP). The MP time becomes quickly dominent because it is performed at a constant rate, and that rate is much slower than the rate of the DPs.

Is this a realistic picture of the world? The single metric, "effective instruction rate", is equivalent in determining the performance of a database machine to the corresponding "million instructions per second" in conventional computers. Neither reflects the complete performance of the machine: I/O design and bandwidth, software design, and so on, must be taken into account. However, "millions of instructions per second" provides a basic "rule of thumb" comparison of different conventional computers. In that the

effective instruction rate shows that the overhead, the time that cannot be multi-processed in special-purpose data manipulation processors, can degrade the performance of a data base machine, it correctly reflects reality.

4. Application Comparisons

The following discussion centers around two points: the number of possible active DPs and the amount of overhead that must be performed by the MP for different designs and different applications. The arguments are based on measured queries where appropriate, but usually appeal to intuitive insight.

Overhead includes the data management functions such as getting the query from the user or host, sending data back, performing concurrency control, protection, etc. Additionally, in a database machine the overhead includes scheduling multiple processors and performing the operating system functions of buffer, disk, and process management, etc.

We define three generic application types: the business system, the bibliographic search system, and the statistical analysis system. These names are used to aid intuitive categorization, and do not imply that these are the complete set of application types.

4.1. Bibliographic Search

The bibliographic search system is characterized by long sweeps through data, with little data produced as the end result, and little analysis of the produced data. These

are search-intensive systems, where the data is not well-structured for the query either because the access paths are not previously well known, so the data cannot be structured for the access, or because the maintenance of the structured access (i.e. indices) is too space or time consuming. Examples of bibliographic search applications are telephone lookups (searching a telephone directory for "sounds like Moore on 10th Street"), literature abstract lookups, and most library applications. With respect to relational systems, these applications are satisfied by the restriction and projection applications of a database management system (DBMS).

Since bibliographic search applications require long reads of relations, the overhead is small compared to the total amount of work in the usual transaction. Therefore, the time spent in the DPs can be assumed to be in the high part of Figure 5, with the percent of DP time at .9 or greater. Therefore, the single-DP system is clearly not the optimal system. The differences in effective instruction rate in this end of the curve are highly sensitive to the proportion of time in the MP; thus, if we can show that any of the remaining systems will have less management tasks, it will necessarily be the optimal system.

To perform the restriction/projection function the bibliographic application requires, the cache-based systems must first bring the data into the cache while the disk-based system can perform the restriction/projection first,

then bring the needed data into the cache. Therefore the cache-based MP must perform more of the management functions of scheduling disk operations, buffer assignment, etc. than the disk-based MP. In a hybrid system, the cache-based DPs would be idle since all the restriction/projection functions can be performed in the disk-based DPs. Therefore, the number of active DPs in a hybrid system would be the same as in the disk-based system, yet the complex architecture would require a higher management overhead. Therefore the disk-associated multiple DP system is best for the bibliographic search application.

4.2. Business Applications

The business system is characterized by short queries to highly structured data. That is, since the query access paths are usually regular, the data can be stored such that it is structured for those access paths (e.g., hashed on account number, or with a B-tree on employee name). Business systems are common: banking applications, inventory, control, credit, accounting, and customer information systems are all examples of the generic type "business systems".

A measured query in the INGRES data management system [HAWT79] that corresponds to queries in the business application performed data manipulation tasks 8.1% of the time. The INGRES implementation was extraodinarily high in overhead due to the particular implementation of INGRES, but it

is reasonable to assume that a query that uses a structured access method (through a hash function or a B-tree index) will spend little time actually manipulating the requested data. Therefore it is clear that for business applications the overhead is at least equal to the data manipulation time.

Note that the architectural simplicity of a single DP system allows the DP to be associated with the MP cache as well as with data streaming in from the disk. This means that the DP can be used in conjunction with the MP to perform overhead tasks. Processes such as checking for protection violations and performing concurrency control can use a special-purpose data manipulation processor to perform specified tasks.

Disk-associated DP systems will perform the same or worse for this application than single-DP systems. In SIMD multiple disk-associated DP systems, a business application results in a SISD system because all the DPs perform the same command; if the data resides on only one partition of the disk (track) only one DP can be active and the rest idle. Therefore a SIMD disk-associated DP system can have no better performance than the single DP system. In MIMD systems the management task (therefore overhead) is increased, which, since the the curve in Figure 5 is below .5, may result in the MIMD system having worse performace than the single-processor system. Furthermore, in any disk-associated DP system the DPs are unavailable to assist

with the overhead tasks. Therefore, the disk-associated DP system has a higher amount of MP work than the single DP system and may perform worse.

Cache-associated processors are usually tightly coupled to the data pages, as in both DIRECT and RAP. Therefore they cannot be used to process the overhead. Furthermore, the business application queries fall on Figure 5 between the 0 and .5 marks, so an increase in the management functions of the machines may force the effective instruction rate of the multiple-DP machine to be less than that of the single DP machine.

The hybrid system can use DPs to perform management functions, as the single DP system does. However, the extra management functions incurred by the complex structure of a hybrid system may result in an increase in MP time. Additionally, Figure 5 indicates that the performance of a multiple DP system is so close to the single-DP system that the additional cost of the hybrid system may not be justified by its performance. Therefore, the single DP system is the best system for the business application.

Business applications reference little data because data is indexed and large searches are unnecessary. It is often suggested that multiple-DP systems (which in effect are associative search systems) make the maintenance and use of indices unnecessary, therefore making all business applications essentially bibliographic search applications. But it must be pointed out that business applications make use

of indices because they are inherently efficient; to locate a record with a B-tree, for example, is an O(log n) operation. To perform the same operation without indices requires a search of the relation, an O(n) operation. The use of associative search techniques reduces the full search time by at most 1/k, where k is the number of DPs used. In true associative search processes, k = n and the O (n/k) operation is better than the O (log n) operation. But for large n (where n is the number of tuples, n = 500 million is common), k = n DPs are impossible to obtain, therefore k << n, and O(log n) is always smaller than O(n/k) Therefore those operations which can use a single DP and indices will profit by doing so.

4.3. Statistical Analysis Applications

Statistical analysis applications are characterized by the correlation of large amounts of data. Databases maintained for statistical analysis purposes include econometric databases (national sales figures, productivity indices, etc.), medical databases (frequency and locations of various illnesses, causes of deaths, etc.), summary census databases (population, agriculture, manufacturing summary statistics), and environmental databases (pollution monitoring results). Statistical analyses of these databases involves correlation of data across several relations, since the analyses often involve factors that appeared unrelated when the data was collected and stored. In a relational system, these appli-

cations are satisfied by the restriction, projection and join functions of the DBMS.

In [HAWT79] an INGRES query was measured that is equivalent to those that occur in statistical analysis applications. 99.4 % of the work was in the data manipulation processes. This is because the join was over two large relations, and required a great deal of processing. Since, in the general case, the statistical analysis applications fall in the high end of Figure 5, the single-DP system is not optimal for this application.

The difference between the statistical analysis and bibliographic applications is that the statistical analysis applications require more joins. Therefore, the evaluation of the effectiveness of the disk-associated multiple-processor machines for the statistical application resolves to an evaluation of their effectiveness in performing joins. A complete, analytical exploration of the differences in join algorithms for database machines is outside the range of this paper, and exists in [BORA80]. However, [BORA80] does not directly address the problem of disk-associated multiple processors, and we desire an approximation for the performance of such a machine in the statistical analysis application. The following is an intuitive argument to develop that estimation.

In a "complete join" the target list contains attributes from both joining relations, while the target list in a "partial join" contains attributes from only one of the

relations. Complete joins, and all joins that are not equi-joins, require re-referencing the joined relations in order to match up qualifying tuples. For instance, consider the query

```
retrieve (emp.name, emp.totsales, div.name)
        where emp.totsales > div.indquota
```

which says to display the employee names and total sales, and division names, for all employees where the employee's total sales was greater than the division's individual sales quota. This is a complete join; the restricted projection of the division and employee relations can be made, but at some point it is necessary to match the employee tuples with the division tuples and associate the correct employee name, totsales with the correct indquota. This type of re-referencing is best performed in a cache system, where the system is not constrained to re-read disk tracks.

Linear join algorithms are used in two disk-associated DP systems (CASSM and CAFS) and also explored in [SHAW80] but these algorithms are linear only for partial, equi-joins because they rely on hashing the joined values to bit-maps and matching tuples based on the double marking of the bit-maps. Therefore, in general, joins perform better in cache-based DP systems than in disk-based systems.

However, joins perform best in hybrid systems. In a hybrid system, the disk-associated DPs can aid in processing joins by the application of algorithms such as the linear

join, and by providing restriction and projection functions. Then the multi-processing of the join functions would take place in the cache. The more complex system requires more managing of the multiple processors, but that is offset by having more active DPs than the other systems (the disk-based DPs performing restriction/projection, the cache-based DPs performing joins) and by the decrease in buffer management and disk-associated "operating system" management functions that having the restriction/projection operations performed in the disk-based DPs provides. Therefore, the hybrid system is the best for staistical analysis applications.

5. Conclusion

We have shown that the best design for the organization and number of data manipulation processors is application dependent. In particular, it was stated that the single-DP organization is best for business applications; that the disk-associated multiple DP organization is best for bibliographic search applications; and that the hybrid system is best for statistical analysis applications.

The comparison was based on a single metric, the effective instruction rate. This is a metric that accounts only for processing functionality, and does not describe the I/O implications of the different organizations of multiple DP systems. It also does not evaluate any of the other differences in database machine performance that are dependent on

the number and organization of the processors, such as algorithms for performing aggregate functions [BORA80]. Nevertheless, in showing that the optimal organization of the processors is application dependent even under a simple metric, we have shown that there needs to be a family of database machines. Conventional computers vary in design according to the target application type ("market"). Likewise, database machines can be designed for specific target application types.

This analysis has lead to interesting insights in the importance of decreasing or multiprocessing the overhead time for queries. Further work is needed to determine processing requirements for overhead functions such as buffer management, concurrency control, MIMD processor management, and so on. Algorithms can be developed to exploit multiprocessing capabilities to decrease the time for the functions that contribute heavily to the overhead. We have shown that database machine performance may be highly dependent on the impact of the managing processor functions, and that much work remains in the characterizing and minimizing that overhead.

BIBLIOGRAPHY

[BABB79] Babb, E., "Implementing a Relational database by Means of Specialized Hardware," ACM Transactions on Database Systems, Vol. 4, No. 1, March 1979, pp.1-29.

[BAN78A] Banerjee, J., and D.K. Hsiao, "Performance Study of a Database Machine in Supporting Relational Databases," Proc. Fourth International Conference on VLDB, 1978.

[BAN78B] Banerjee, J., and D.K. Hsiao, "The Use of a 'Non-Relational' Database Machine in Supporting Relational Databases," Proc. Fourth Workshop on Computer Architecture for Non-numeric Processing, Syracuse, Aug. 1978.

[BANC80] Bancilhon, F., "VERSO: A Backend Relational Machine," Proc. Sixth VLDB, 1980, p.394.

[BAUM76], Baum, Richard I., D. K. Hsaio and K. Kannan, "The Architecture of a Database Computer - Part I: Concepts and Capabilities," National Technical Information Service Number AD-A034 154

[BORA79] Boral, H., and D.J. DeWitt, "Processor Allocation Strategies for Multiprocessor Database Machines," Computer Sciences Technical Report No. 368, University of Wisconsin, Oct. 1979.

[BORA80] Boral, H., D.J. DeWitt, D. Friedland and W.K.

Wilkenson, "Parallel Algorithms for the Execution of Relational Database Operations" Computer Sciences Technical Report no. 402, University of Wisconsin-Madison, October, 1980.

[BRIT80] Britton-Lee, Inc. "IDM 500 Intelligent Database Machine Product Description,", Britton-Lee Inc., 90 Albright Way, Los Gatos 95030.

[COUL72] Coulouris, G.F., Evans, J.M., and R.W.Mitchell, "Towards Content Addressing in Databases," Computer Journal, Vol. 15, No. 2, 1972, pp. 95-98.

[DEWI78] Dewitt, D.J., "DIRECT - A Multiprocessor Organization for Supporting Relational Data Base Management Systems," Proc. Fifth Annual Symposium on Computer Architecture, 1978.

[DEWI79] Dewitt, D.J., "Query Execution in DIRECT", Proceedings of the ACM-SIGMOD 1979 International Conference on Management of Data, May 1979, pp 13-22.

[EPS80A] Epstein, Robert and Paula Hawthorn, "Aid in the '80s," Datamation, Feb. 1980, pp. 156-158.

[EPS80B] Epstein, Robert and Paula Hawthorn, "Design Decisions for the Intelligent Database Machine", Proceedings 1980 NCC, AFIPS Vol. 49, pp.237-241.

[HAWT79] Hawthorn, Paula and M. Stonebraker, "Performance

Analysis of a Relational Database Management System," Proceedings of the ACM-SIGMOD 1979 International Conference on Management of Data, May 1979, pp 1-12.

[HAWT80] Hawthorn, Paula, and D. J. Dewitt, "Performance Analysis of Alternative Database Machine Architectures," Computer Sciences Technical Report no. 383, University of Wisconsin-Madison, March 1980.

[HSIA79] Hsiao, D.K. and J. Memon, "The Post Processing Functions of a Database Computer," Computer and Information Science Technical Report OSU--CISRC-TR-76-2, Ohio State University, 1979.

[HSI76A] Hsiao, D.K., and K. Kannan, "The Architecture of a Database Computer - Part II: The Design of Structure Memory and its Related Processors," National Technical Information Service Number AD/A035 178

[HSI76B] Hsiao, D.K., and K. Kannan, "The Architecture of a Database Computer - Part III: The Design of Mass Memory and its Related Components," National Technical Information Service Number AD-A036 217

[KANN78] Kannan, K., "The Design of a Mass Memory for a Database Computer," Proc. Fifth Annual IEEE Symposium on Computer Architecture April 1978.

[LIPO78] Lipovski, G. J., "Architectural Features of CASSM: a Context Addressed Segment Sequential Memory", Proc.

Fifth Annual IEEE Symposium on Computer Architecture, April 1978.

[OZKA75] Ozkarahan, E.A., S.A. Schuster, and K.C. Sevcik, "RAP - Associative Processor for Database Management," AFIPS Conference Proceedings, Vol. 44, 1975, pp. 379 - 388

[OZKA77] Ozkarahan, E.A., S.A. Schuster, and K.C. Sevcik, "Performance Evaluation of a Relational Associative Processor," ACM Transactions on Database Systems, Vol. 2, No. 2, June 1977.

[ROSE77] Rosenthal, R.S., "The Data Management Machine, A Classification," ACM Sigir-Sigarch-Sigmod Third Workshop on Computer Architecture, Syracuse, New York, pp. 35-37.

[SADO78] Sadowski, Paul W. and S.A. Schuster, "Exploiting Parallelism in a Relational Associative Processor,", Proceedings Fourth Workshop on Computer Architecture for Non-Numeric Processing, August, 1978.

[SCHU78] Schuster, S.A., et. al., "RAP.2 - An Associative Processor for Data Bases", Proceedings, Fifth Annual IEEE Symposium for Computer Architecture, April, 1978.

[SHAW80] Shaw, David E., "A Relational Database Machine Architecture", Proceedings Fifth Workshop on Computer Architecture for Non-Numeric Processing, March, 1980.

[SMIT79] Smith, Diane C.P., and J.M. Smith, "Relational Data Base Machines," IEEE Computer, March 1979, pp. 28-38.

[SU75] Su, S.Y.W. and G.J. Lipovski, "CASSM: A Cellular System for Very Large Data Bases", Proceedings of the VLDB, 1975, pp. 456-472.

[SU79] Su, S.Y.W., "Cellular-Logic Devices: Concepts and Applications", IEEE Computer, March 1979, pp. 11-28.

ACKNOWLEDGMENT

This work was supported by the Applied Mathematics Sciences Research Program of the Office of Energy Research of the U.S. Department of Energy under Contract No. W-7405-ENG-48.

This report was done with support from the Department of Energy. Any conclusions or opinions expressed in this report represent solely those of the author(s) and not necessarily those of The Regents of the University of California, the Lawrence Berkeley Laboratory or the Department of Energy.

Reference to a company or product name does not imply approval or recommendation of the product by the University of California or the U.S. Department of Energy to the exclusion of others that may be suitable.

TECHNICAL INFORMATION DEPARTMENT
LAWRENCE BERKELEY LABORATORY
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720