PERFORMANCE ANALYSIS OF DISTRIBUTED DATA BASE SYSTEMS

by

Michael Stonebraker, John Woodfill, Jeff Ranstrom,
Joseph Kalash, Kenneth Arnold and Erika Andersen

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
UNIVERSITY OF CALIFORNIA
BERKELEY, CA.

ABSTRACT

In this paper we briefly present the design of a distributed relational data base system. Then, we discuss experimental observations of the performance of that system executing both short and long commands. Conclusions are also drawn concerning metrics that distributed query processing heuristics should attempt to minimize. Lastly, we comment on architectures which appear viable for distributed data base applications.

* Research Sponsored by the National Science Foundation under Grant MCS-8211528 and by the Defense Advanced Research Projects Agency under Contract N00039-C-0235.

1. INTRODUCTION

Many algorithms have been proposed to solve distributed relational data base problems in the areas of:

a) distributed concurrency control

b) distributed crash recovery

c) support of multiple copies of data

d) distributed command processing

There is currently little quantitative knowledge on the performance of such algorithms. Previous work has been based exclusively on simulation, *e.g.* [RIES79, GARC79a, GARC79b, LIN81] or formal modeling, *e.g.* [GELE78, BERN79]. One of the objectives of this research is to provide empirical results concerning the performance of various algorithms.

This paper first presents a short description of a working prototype distributed data base system. Then, we present the results of a collection of experiments on this prototype. Conclusions concerning query processing algorithms are drawn as appropriate. Lastly, comments on viable distributed architectures for data base applications are presented.

2. DISTRIBUTED INGRES

Distributed INGRES operates on a collection of DEC VAX 11/780s and 11/750s connected by a 3 mbit ethernet. All run the 4.1cBSD, a version of the UNIX [RITC75] operating system enhanced at Berkeley with paging, numerous program development tools, remote interprocess communication, and remote process execution.

Most features of Distributed INGRES [EPST78] are currently operational. A master INGRES process runs at the site where the command originated and slave INGRES processes run at each site which have data involved in the command. The master process parses the command, resolves any views, and creates an action plan to solve the command using the fragment and replicate technique. The slave process is essentially single-machine INGRES [STON76, STON80] with minor extensions and with the parser removed. The coordinator and slaves communicate using the 4.1cBSD interprocess message system.

Distributed INGRES supports fragments of relations at different sites. For example, one can distribute the relation

```
EMP (name, salary, manager, age, dept)
```

as follows:

```
range of E is EMP
distribute E
  at Berkeley where E.dept = "shoe"
  at Paris    where E.dept = "toy"
  at Boston   where E.dept != "toy" and
                    E.dept != "shoe"
```

Berkeley, Paris and Boston are logical names of machines which are mapped to site addresses by a lookup table. A single site relation is a special case of the distribute command, *e.g.*

```
distribute ONE-SITE at Berkeley
```

Currently, all QUEL commands without aggregates are processed correctly for distributed data. Consider, for example, the following update:

```
range of E is EMP
replace E(dept = "toy") where e.salary > 10000
```

This command will be processed by all sites containing fragments of the EMP relation. All qualifying tuples are updated and their site location may be changed. For example, the tuple of an employee earning more than 10000 in the shoe department would be moved from Berkeley to Paris.

Distributed INGRES uses a two phase commit protocol [GRAY78, LAMP76]. Slaves send a "ready" message to the master when they are prepared to commit an update. Tuples which change sites are included with this message. The master then redistributes the tuples by piggybacking them onto the commit message. A three phase commit protocol can optionally be used [SKEE82] for added reliability. In this case the above redistribution is handled in the second phase.

When a command spans data at multiple sites, a rudimentary version of the "fragment and replicate" query processing strategy is used. For example, suppose a second relation

```
DEPT (dname, floor, budget)
```

exists at two sites as follows:

```
distribute D
  at Berkeley where D.budget > 5
  at Paris    where D.budget <= 5
```

Consider the query submitted by a Boston user:

```
range of E is EMP
range of D is DEPT
retrieve (E.name) where E.dept = D.dname
                  and D.floor = 1
```

First, the one variable clause "D.floor = 1" is detached and run at both Berkeley and Paris, *i.e.*

```
range of D is DEPT
retrieve into TEMP (D.dname) where D.floor = 1
```

The original query now becomes

```
range of E is EMP
range of D is TEMP
retrieve (E.name) where E.dept = D.dname
```

To satisfy the query, data movement must now take place. One relation (say TEMP) is replicated at each processing site. Hence, both Berkeley and Paris send their TEMP relations to each site which has a fragment of EMP. Therefore, the needed transmissions are:

```
TEMP(Paris) -> Boston
TEMP(Paris) -> Berkeley
TEMP(Berkeley) -> Paris
TEMP(Berkeley) -> Boston
```

Now, all three sites have a complete copy of TEMP and a fragment of the EMP relation. The above query is now performed at each site, and the resulting tuples are returned to the master site, where they are displayed to the user.

Since our ETHERNET has the hardware capability to support broadcast, it is possible to perform the above four transfers by broadcasting each fragment of TEMP. However, the 4.1cBSD operating system does not support multicast or broadcast transmissions. Consequently, the above four transmissions occur separately, and the strategy of replication may perform poorly [EPST78]. The network on which we planned to run [ROWE79] supported broadcast, and we have not subsequently modified the query processing heuristics.

At the moment, the relation to be replicated is chosen arbitrarily, so TEMP and EMP are equally likely to be selected for movement. A more elegant strategy is being planned.

3. SIMPLE UPDATES

In all experiments we use the EMP and DEPT relations as discussed in Section 1. Our data base consists of 30,000 EMP tuples, each 38 bytes long and 1500 DEPT tuples each 18 bytes long. In all cases we will be comparing the performance of Distributed and single-site INGRES.

The first benchmark consists of 1000 random updates of the form:

```
replace E (salary = K) WHERE E.name = L
```

The n-site data base was distributed as follows:

```
distribute E
at site-a where e.name < j1
atsite-b where e.name >= j1
          and e.name < j2
            .
            .
            .
atsite-n where e.name >= jn−1
```

The constants $j_{1}, \ldots, j_{n-1}$ were chosen so that exactly 1000/*n* updates were directed to each site. The number of sites, *n*, was varied from 1 to 3.

Each site ran a script which contained 1000/*n* updates and processed the next command when it received "done" from the previous one. In this way there was a master INGRES at each site and we avoided creating a bottleneck at a single coordinating site.

Note that this benchmark consists of a large collection of small transactions, each of which can be completely processed at a single site. A distributed data base should perform well in this situation.

Table 1 indicates for each configuration the CPU time spent inside the operating system, the CPU time spent inside the INGRES code and the elapsed time. The benchmark was run on a VAX 11/780 along with 0, 1 or 2 VAX 11/750's. Unfortunately, the 11/750's have varying amounts of main memory, disk systems, and buffer space allocations. Moreover, the error rate of network transmission varies between pairs of machines. As a result, a fair amount of random variation of the numbers must be expected.

For the distributed processing configurations, the reported times are a sum of the time spent by the master INGRES at that site along with the times spent by any slave INGRESs on behalf of masters at other sites. According to local benchmarks, an 11/750 is about 0.629 times as fast as an 11/780 [HAGG83]; hence total CPU time is calculated by scaling 11/750 time by the above factor and is reported in the row labeled by n*780.

Several conclusions can be drawn from these results. First, Distributed INGRES is about 20 percent slower than normal INGRES when run on a local data base. Distributed INGRES must check the distribution criteria to ascertain that each of the commands is a local one. Currently, this checking is performed at run time; however, for better performance it could be performed at compilation time. In addition, each updated tuple must also be checked against the distribution criteria to ensure that it does not change sites (i.e. that the dept field is not being changed).

Second, Distributed INGRES on a one machine foreign data base is about 10 percent slower than on a local data base. The foreign data requires master INGRES to communicate with a non-local slave instead of a local slave, and this requires extra user and system CPU time.

Third, 2*780 and 3*780 Distributed INGRES use 20 percent more CPU time than Distributed INGRES on a local data base and 45 percent more CPU time than single-site INGRES. Both systems use marginally more CPU time than Distributed INGRES on a one-site foreign data base. The benefit of these configurations is increased parallel processing; hence the benchmark finishes respectively 25 and 40 percent faster. Of course, the benchmark would have finished even faster if the additional machines were

	user time	system time	elapsed time
Normal INGRES			
11/780	7:34	3:04	22:34
Distributed INGRES - local data base			
11/780	9:06	3:53	26:57
Distributed INGRES - foreign data base			
11/750	7:58	3:02	28:37
11/780	5:34	2:57	
02*780	10:35	4:51	
Distributed INGRES - 2 sites			
11/780	5:14	2:24	15:30
11/750	8:24	4:05	16:46
02*780	10:31	4:58	
Distributed INGRES - three sites			
11/780	3:43	1:30	12:43
11/750	5:28	2:16	13:34
11/750	5:48	2:13	13:22
03*780	11:09	4:30	

Performance of Simple Updates
Table 1

11/780s. We suspect that a collection of n 11/780s could finish the benchmark in approximately 28/n minutes.

Lastly, note that the 3 site benchmark uses the same amount of CPU time as the two site benchmark. It is reasonable to expect that the total CPU time would continue to be a constant as additional sites were added. Hence, we predict that total aggregate CPU time would remain a constant as sites are added and would be split among an increasing number of machines.

Benchmark 1 on a foreign data base results in 522,880 bytes being transferred across the network, and less than two percent of the available bandwidth is consumed. It appears that a large number of machines could be added to the ETHERNET before bandwidth limitations arise.

4. ONE RELATION RETRIEVES

In this benchmark we attempted to load the network as fully as possible with the following query:

```
range of E is EMP
retrieve (E.all)
```

The result of this query is 30000 tuples which would ordinarily be printed on the terminal. To stress differences in the environments being tested, we discarded the qualifying tuples in both this benchmark and the subsequent one. Hence, the cost of printing more than 1 mbyte of data is not included in the results presented in Table 2. In the 2 site and 3 site benchmarks the EMP relation is uniformly distributed across the sites. Moreover, we are timing several repetitions of the query submitted from a single job stream and then averaging them.

Distributed INGRES on a local data base runs at about the same speed as single-site INGRES. The extra overhead of discovering that the query is local is amortized over a large amount of processing, so the two systems perform comparably.

On a foreign data base distributed INGRES is 0:49 seconds slower. In this environment, a slave must write the EMP relation into a temporary file and pass it across the network to a another file. Consequently, there are a total of two copies made of the 1200 block EMP relation.

The cost of a executing a remote copy of the 1200 block file is 0:17 of elapsed time and 0:11 of system CPU time. Hence, about 32 percent of the 0:49 difference is explained by the network overhead; the rest is added INGRES overhead. This remote copy consumes about 19.3 percent of the 3 mbit bandwidth. Because INGRES adds extra overhead, it uses only 6 percent of the available bandwidth. Obviously a large number of concurrent data base users would be required before INGRES could use any substantial fraction of the ETHERNET bandwidth.

When the data base is distributed over multiple sites, the total CPU time remains approximately constant and is distributed evenly over the machines. When two sites are present, about 50 percent of the CPU cycles are offloaded to an 11/750 which is 0.629 times as fast. The maximum improvement possible in this

	user time	system time	elapsed time
Normal INGRES			
11/780	1:44	0:15	2:03
Distributed INGRES - local data base			
11/780	1:47	0:10	2:05
Distributed INGRES - foreign data base			
11/750	0:03	0:03	2:54
11/780	1:47	0:20	
02*780	1:49	0:22	
Distributed INGRES - 2 sites			
11/780	1:06	0:19	
11/750	1:36	0:15	2:59
02*780	2:06	0:28	
Distributed INGRES - three sites			
11/780	0:35	0:05	2:37
11/750	1:13	0:16	
11/750	1:12	0:17	
03*780	2:06	0:26	

Performance of One-relation Retrieves
Table 3

configuration is about 25 percent, and it appears that INGRES overhead offsets this gain. With three sites dividing the work, response time begins to improve, and this improvement should continue as new sites are added.

Four conclusions can be drawn from the results of this benchmark and the above discussion. First, query processing heuristics should account for the speed of the various machines when deciding optimal strategies. To achieve minimum response time using our configuration, one should give the 11/780 disproportionately more work than the 11/750s. Second, bandwidth will never be a problem in our environment. Even operating system file transfers do not come close to using the entire bandwidth, and INGRES relations cannot be moved any faster than OS files. Third, data base and file servers are often proposed as useful architectural concepts in a local network environment. However, this configuration is closely approximated by a foreign data base which had the next to worst performance of the ones tested. Unless a server is much faster than other machines on the network or unless other machines do not have disks, the merits of a server seem doubtful. Lastly, it appears desirable to split complex queries among a large number of sites and take advantage of the resulting parallel processing.

5. JOIN EXPERIMENT

The last experiment executed the natural join of EMP and DEPT, with EMP hashed on the dept field and DEPT hashed on the dname field, i.e:

```
range of E is EMP
range of D is DEPT
retrieve (E.all, D.all) where E.dept = D.dname
```

The same environments were tested as in the previous sections. In the 2 and 3 site cases both EMP and DEPT were uniformly distributed, and DEPT was selected as the relation to be replicated in query processing. Table 4 contains the measured results.

The elapsed time for the foreign data base is longer than expected by about 2-3 minutes; otherwise these numbers are very similar to the preceding two data sets. Hence, we will not comment on their relative magnitudes but rather discuss other points.

First, the two and three site versions moved the DEPT relation to solve the query. We forced distributed INGRES to instead move the EMP relation, and the results were about 20 times slower than those reported. The explanation is somewhat subtle. When Distributed INGRES replicates a relation at multiple sites, it loses the access structure of the relation involved and does not recreate the original access path for the composite relation. Hence, if DEPT or EMP is moved, it becomes a heap at each site. Local INGRES algorithms solve the join by iterating over the smaller of the two relations, in this case DEPT. If DEPT is moved, then INGRES will iterate over a heap producing a large collection of queries of the form:

```
retrieve (E.all, -constants- )
   where E.dept = constant
```

	user time	system time	elapsed time
Normal INGRES			
11/780	8:41	0:38	9:37
Distributed INGRES - local data base			
11/780	8:57	0:47	10:34
Distributed INGRES - foreign data base			
11/750	0:05	0:03	14:32
11/780	9:01	0:42	
02*780	9:04	0:44	
Distributed INGRES - 2 sites			
11/780	4:28	:21	10:45
11/750	7:56	1:02	
02*780	9:28	1:00	
Distributed INGRES - three sites			
11/780	3:11	:13	7:41
11/750	5:27	:43	
11/750	5:14	:40	
03*780	9:54	1:05	

Performance of Joins
Table 4

These queries can then be executed by a hashed access to the EMP relation. However, if EMP is moved and becomes a heap, a large number of queries are generated, each requiring a complete scan of the EMP relation.

We did not execute the query with EMP at one site and DEPT at another. In this case the query processing module should move the DEPT relation to the site of EMP. This should add only a few seconds of overhead to the distributed INGRES times for a local data base.

We also did not force the obvious semi-join strategy indicated by the following commands.

```
retrieve into W(E.dept)
move W
```

```
retrieve into W2 (D.all) where D.dname = W.dept
move W2
retrieve (E.all, W2.all) where E.dept = W2.dname
```

Since all values of dname appear in the EMP relation, W2 is exactly the size of DEPT. This algorithm will consequently perform more poorly than all other ones since it will perform a projection of the EMP relation in addition to the work done by other algorithms. Given that bandwidth is not a consideration in our environment, semi-joins, which must execute the query twice, will seldom be advantageous.

6. CONCLUSIONS

This paper presented timings for a distributed data base system. By and large, they are extremely encouraging. Although Distributed INGRES is not highly optimized, it does not add a large amount of overhead. It is expected that judicious tuning could make it competitive with single-site INGRES on local data bases. On distributed data, the costs of moving data are not excessive and result in substantial parallelism.

REFERENCES

[BERN79] Bernstein, P. and Chiu, D., "Using Semi-joins to Solve Relational Queries", Computer Corp. of America, Cambridge, Mass., Jan. 1979.

[EPST78] Epstein, R., et. al., "Distributed Query Processing in a Relational Data Base System," Proc. 1978 ACM-SIGMOD Conference on Management of Data, Austin, Texas, May, 1978.

[GARC79a] Garcia-Molina, H., "Centralized Control Update Algorithms for Fully Redundant Distributed Data Bases," Proc. 1st International Conference on Distributed Computing, Huntsville, Ala., Oct. 1979.

[GARC79b] Garcia-Molina, H., "Performance of Update Algorithms for Replicated Data in a Distributed Data Base," PhD Thesis, Stanford University, Computer Science Dept, June 1979.

[GELE78] Gelenbe, E. and Sevcik, K., "Analysis of Update Synchronization for Multiple Copy Data Bases," Proc. 3rd Berkeley Workshop on Distributed Data Bases and Computer Networks, San Francisco, Ca., February 1978.

[GRAY78] Gray, J., "Notes on Data Base Operating Systems," in Operating Systems: An Advanced Course, Springer-Verlag, 1978, pp393-481.

[HAGG83] Hagmann, R., private communication

[LAMP76] Lampson, B. and Sturgis, H., "Crash Recovery in a Distributed System," Xerox Palo Alto Research Center, 1976.

[LIN81] Lin, W., "Performance Evaluation of Two Concurrency Control Mechanisms in a Distributed Data Base System," Proc. 1981 ACM-SIGMOD Conference on

Management of Data, Ann Arbor, Mich., May 1981.

[RIES79] Ries, D., "The Effects of Concurrency Control on Data Base Management System Performance," Electronics Research Laboratory, Univ. of California, Memo ERL M79/20, April 1979.

[RITC75] Ritchie, D. and Thompson, K., "The UNIX Timesharing System," CACM, June 1975.

[ROWE79] Rowe, L. and Birman, K., "Network Support for a Distributed Data Base System", Proceedings of the Fourth Berkeley Workshop on Distributed Data Management and Computer Networks, August, 1979, San Francisco, California.

[SKEE82] Skeen, D., "A Quorum-Based Commit Protocol," Proc. 6th Berkeley Workshop on Distributed Data Bases and Computer Networks, Pacific Grove, Ca., Feb 1982.

[STON76] Stonebraker, M. et. al., "The Design and Implementation of INGRES," TODS 2, 3, September 1976.

[STON80] Stonebraker, M., "Retrospection on a Data Base System," TODS, March 1980.