MEASUREMENTS OF DATA SKEW IN TWO DATABASES

Christopher B. Walton, Matt L. Pinsonneault,
and Furman Haddix

Department of Computer Sciences
The University of Texas at Austin
Austin, Texas 78712-1188

TR-90-32 October 1990

ABSTRACT

Data distributions are presented for relations in two databases: stock trading data and message traffic in a military communications system. This report makes two research contributions. Formal definitions of skew parameters are added to the relative partition model of data skew. Finally, although the observed databases reside on a single node system, skew parameters for three types of data skew are estimated for a worst case partitioning.

Keywords: Data skew, relational databases, parallel joins, relative partition model.

1 Introduction

In recent years, architectures and algorithms for parallel joins have been popular topics for research. On class of architectures is multicomputers: homogeneous computational nodes that communicate by message passaging. This also known as the shared-nothing architecture[24]. These systems typically act as back ends, servicing queries that are formulated on a host system

Specific parallel join algorithms have been proposed in [12,8,2,1,5,29]; while more general analyses and comparisons are presented in [21,3,13,10,14,23]. However, nearly all of this work assumes that data is uniformly distributed among processors at every stage of the join. Under these idealized conditions, join algorithms are quite scalable [5,6,9]. That is, performance increases approximately in proportion to the number of processors.

In reality, data skew – irregular distribution of data – is present in most cases. The frequent existence of data skew is documented in [17,27]. In response to this recognition, skew resistant algorithms (algorithms that are designed to maintain good performance even if data is skewed) have been proposed for partitioning [11], sorting[16], merging [25], and joins[29,19,28].

While algorithm design is a valuable activity, it does not provide a deeper understanding of the phenomenon of data skew itself. Current models of data skew have two major shortcomings. First, although most research performed to date regards skew as a homogeneous phenomenon, it has many different causes and manifestations. To the best of the authors' knowledge, the types of data skew presented in [27,26] are the first attempt to classify skew.

Another problem is quantifying data skew. Many authors favor models based on the Zipf[30] distribution [7,17,22]. Simplier models are proposed in [13,14,27]. However, there have been few efforts to apply these models to actual data bases; limited examples are found in [4,18,20,28].

This report makes two contributions. First, it extends the relative partition model[27] by providing formal definitions for each type of skew. These definitions are then used to quantify selected skew parameters in two actual databases.

The remainder of the report is organized as follows. Section 2 provides formal definition of skew parameters. Section 3 presents data distributions for a stock trading database, as well as estimates of skew parameters. Section 4 makes a similar presentation for military communications data.

2 Measuring skew parameters from data

This section extends previous work [27,26] by presenting formal definitions of the skew parameter Q for each type of data skew. The four types of skew are presented in the order in which they would occur during the processing of a query: tuple population skew, selectivity skew, hash partition skew, and finally join probability skew. Note that skew parameters for each type are defined so as to be unaffected by other types of skew. For example, the skew parameter for selectivity skew is not affected by tuple population skew.

Let $||S||$ and $||R||$ be the cardinalities of relations S and R respectively. Let $\mathcal{P}$ be a partioning function that splits tuples among N nodes. The set of tuples in S stored at node i is denoted by $\mathcal{P}_i(S)$.

2.1 Tuple Population Skew (TPS)

The skew parameter Q is the ratio between the largest per-node cardinality and the average node cardinality. That is:

$$Q^{TPS} = \frac{\max_i \{ \, ||\mathcal{P}_i(S)|| \, \}}{||S||/N} = \frac{N \max_i \{ \, ||\mathcal{P}_i(S)|| \, \}}{||S||} \tag{1}$$

2.2 Selectivity Skew (SS)

For SS, Q indicates the ratio between the largest fraction of tuples remaining after local selection and the average fraction remaining. Let L_i be the local selection predicate at node i, then $L_i(\mathcal{P}_i(S))$ denotes the set of tuples at node i that satisfy the selection predicate. The local selectivity σ_i is defined as:

$$\sigma_i = \frac{||L_i(\mathcal{P}_i(S))||}{||\mathcal{P}_i(S)||} \tag{2}$$

The selectivity skew is then:

$$Q^{SS} = \frac{N \max_i \{ \, \sigma_i \, \}}{\sum_{i=1}^{N} \sigma_i} \tag{3}$$

Note that if tuple population skew is not present, $||\mathcal{P}_i(S)||$ is the same for all nodes, and equation 3 reduces to:

$$Q^{SS} = \frac{N \max_i \{ ||L_i(\mathcal{P}_i(S))|| \}}{||S||} \tag{4}$$

2.3 Hash Partition Skew (HPS)

Hash partition skew occurs when there is a mismatch between the distribution of join key values in a relation and the distribution expected by the hash function. It has two different causes. **Join key skew** occurs when join key values are not uniformly distributed among tuples; it is a property of the data. In contrast, **Hash function skew** is a property of the hash function. It occurs when the range of the hash function (bucket numbers) is less uniformly distributed than the domain (hash key values).

In either case, HPS is modeled by assuming the Q-node holds Q times as many tuples after the hash partition phase. See [26] for a more extensive discussion and examples.

Let $\mathcal{H}(S)$ be a partitioning hash function (the same at all nodes). $\mathcal{H}_i(S)$ is the set of tuples hashed to node i, regardless of where the tuples were initially stored. To avoid confusion with S, the relation before local selection, we define:

$$S_i^* \equiv \bigcup_{j=1}^{N} \mathcal{H}_i(L_j(\mathcal{P}_j(S))) \tag{5}$$

The definition for R_i^* is similar. Given this definition, the skew parameter becomes:

$$Q^{HPS} = \frac{N \max_i \left\{ \|S_i^*\| \right\}}{\sum_{i=1}^{N} \|S_i^*\|} \tag{6}$$

2.4 Join Probability Skew (JPS)

Join probability skew quantifies how much the cardinalities of join outputs vary between nodes. To adjust for differences in the size of the join inputs, we define the local join selectivity ρ_i.

$$\rho_i \equiv \frac{\|S_i^* \bowtie R_i^*\|}{\|S_i^*\| \, \|R_i^*\|} \tag{7}$$

the skew factor then becomes:

$$Q^{JPS} = \frac{N \max_i \rho_i}{\sum_{i=1}^{N} \rho_i} \tag{8}$$

In the special case where $\|S_i^*\|$ and $\|R_i^*\|$ are the same on all nodes (join inputs are the same size at all nodes), equation 8 reduces to:

$$Q^{JPS} = \frac{N \max_i \left\{ \|S_i^* \bowtie R_i^*\| \right\}}{\sum_{i=1}^{N} \|S_i^* \bowtie R_i^*\|} \tag{9}$$

Note that the semantics of both databases examined in this report are such that the following conditions are true of all pairs of relations that are typically joined:

1. All tuples participate in the join. That is, all join key values occur in both relations.
2. Join key values are duplicated only in the larger relation, while join key values in the smaller relation are unique.

Under these circumstances, every tuple in the larger relation joins with exactly one tuple in the smaller relation. Thus, the size of the join output is determined by the size of the larger R relation; that is:

$$||S_i^* \bowtie R_i^*|| = ||S_i^*||$$

Since the definition of JPS (see equation 7) normalizes over the size of the join inputs, it impossible for ρ_i to vary between partitions if the above relationship holds. Therefore, a reasonable example of JPS cannot be presented here.

3 Stock trading data

3.1 Database Overview

The first system analyzed was a large database of stock trading results The subsystem chosen for study contained trading data for over six thousand companies and 1.5 million trading period records. These records contained such information as share prices, trading volume, and return on investment. The relations of interest were in the section of the database relating to daily NYSE (New York Stock Exchange) and AMEX (American Exchange) stock trading results.

Figure 1 illustrates the illustrates the Database structure; relations are in uppercase and tuple fields in lowercase.

DEL(cuisp)
NAME(cuisp)
HEAD(cuisp,numnam,numdel,begret,endret)
TRAD(cuisp,dcalctr)
CAL(dcalctr)

Figure 1: table structure of stock trading database

More specifically, the use of each relation is as follows

HEAD stores header information for a company. One HEAD record exists per company.

CAL calendar day trade results. Each one has a TRAD record.

TRAD header for CAL record. One exists for each CAL, and usually a large number of TRAD records exist per company.

NAME list of company names; a company may have different corporate names at different times.

DEL a list of companies deleted from the database.

While the significant fields in these records are:

cuisp an eight character company identifier.
numnam number of company names.
numdel number of deleted records.
dcalcntr integer pointer to a calendar day record.
begret beginning return number (see below).
endret final return number (see below).

3.2 Distribution Analysis

According to the database administrator, over 90 percent of all joins performed in queries to the database involve the TRAD and CAL records[15]. Since there is a one-to-one correspondence between TRAD and CAL, we could extract the distribution of records merely by determining how many TRAD records exist per company.

Each CAL record is assigned a return number, which is unique and contiguous for each company. Return numbers ascend continuously and are not reused. Each HEAD record has two fields (**Begret** and **Endret**) that indicate the starting and ending return numbers for a company. Since the return numbers are guaranteed to be sequential, with no omissions, the difference between **Endret** and **Begret** will yield the number of TRAD records that exist for each company. An SQL query was written and executed to perform this calculation, yielding the number of TRAD records for each company.

Once the data was obtained, the following quantities were calculated: arithmetic mean, standard deviation, low and high values, most frequent occurrence, a histogram of distributions, and a detailed distribution list sorted by number of TRAD records per company. Figure 2 shows the numbers of companies whose number of TRAD records falls into some interval. (each line represents an interval of 50, e.g. 1000 to 1050 TRAD records). Note that the horizontal axis does not represent the number of trades, as the intervals have been sorted by frequency.

As can be seen from the histograms in figure 2, the results seem to conform fairly well to a Zipf distribution. The largest accumulation is companies with 6,662 TRAD records (600 companies). There are several explanations for this – perhaps these were the original companies when the database was started, or prehaps only 6,662 days' trade data is kept on-line.

3.3 calculation of skew parameters

Since these data were obtained from a single node system, tuples must be remapped to a hypothetical multicomputer system in order to estimate data skew. That is, measurements of skew factors cannot be made directly. Instead, some partitioning scheme must be assumed. The stock trading data will be used to illustrate TPS and SS.

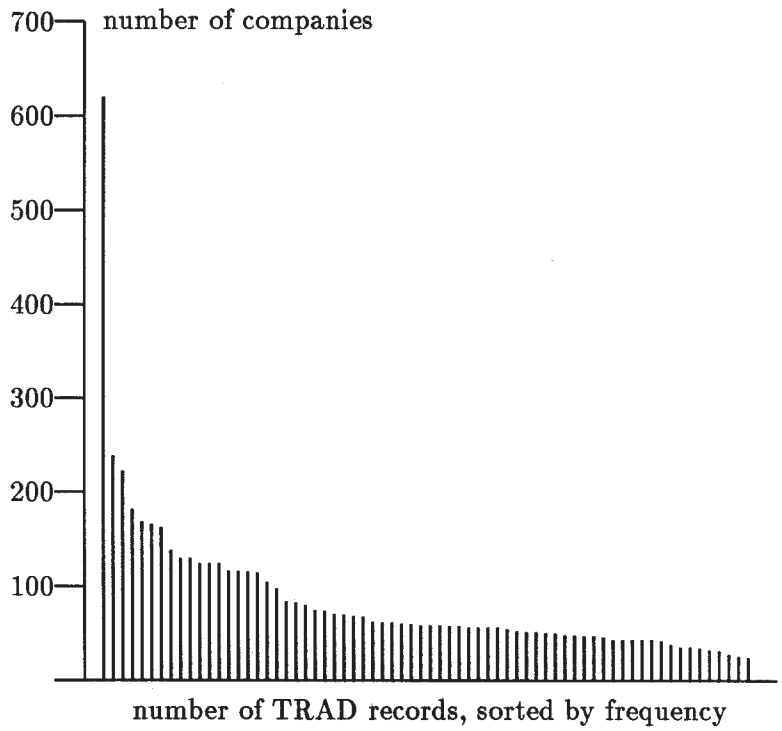

Figure 2: data distribution in stock trading database

3.3.1 tuple population skew

For a worst case analysis of tuple population skew, it was assumed that each node stored an equal number of companies. That is, if there are N nodes (or partitions), then each one holds the tuples associated with $1/N$ of the companies. Note that the oldest companies will have the most trading period (TRAD records). In the worst case, the $1/N$ oldest companies will be concentrated on the same node. Under this scenario, $Q^{TPS} \approx 2.3$ for $N = 8$ and $N = 64$.

3.3.2 selectivity skew

Recall that the extent of selectivity skew is determined by the nature of the local selection query. To evaluate selectivity skew, it was arbitrarily assumed that local selection eliminated all companies with less that 5000 TRAD records. This could represent a query to select all companies with some minimum age. Assuming this selection predicate and the initial distribution described above, Q^{SS} is 4.4 for $N = 8$ and 4.5 for $N = 64$.

4 military communications data

The second database examined for this report consists of message traffic for a military communications system designed to coordinate artillery fire. The data reported here were collected during an evaluation of commercial database products for possible incorporation in a Field Artillery Tactical Data System (FATDS).

Tactical Systems are being developed to automate selected command and control functions in order to provide more efficient management of the fire support resources on the battlefield. This is achieved through, reduced reaction time, faster and better use of target information, and more efficient distribution of available firepower to the supported maneuver force. Such systems consist of computers and remote devices linked by radio communications equipment.

Figure 3 shows the relations in one possible database configuration.

MESSAGE(source,destination,net,text)
SUBSCRIBER(address,net,player-num)
FORCE-STRUCT(player-num,device,player-name)
DEVICE-TAB(device,description)

Figure 3: relation structure for Tactical Data System test

Major variables include:

address single character identifier of message source or destination.

net tactical communications network; may be implemented as a hardwired channel or a radio frequency.

player-num numeric identifier of fire support element or tactical unit.

player-name military unit designation for a player.

device type of communications device

4.1 calculation of skew parameters

4.1.1 Join Key Skew

Recall that join key skew (JKS) is one of the two sources of HPS. Tuples in the MESSAGE relation may be joined with the SUBSCRIBER relation; the join key is the concatenation of an address (source or destination) and net. Such a join could be part of a query to find information about the player who sent or received a given message. Table 1 lists the number of messages sent and received by each address. Since the number of tuples varies between join key values, the message traffic distribution provides an example of join key skew.

For example, consider an 8 node system where all messages for each address (regardless of net) are hashed to node. Also assume that messages for devices P,Q, and X are all hashed to node 7.

Table 2 gives the number of tuples in each partition. Under these assumptions Q^{JKS} is 3.5 for message destination and 3.8 for message source.

Address	Destination			Source		
	Net-1	Net-2	Net-3	Net-1	Net-2	Net-3
A	890	338	50	781	320	41
C	2277	803	668	1916	720	769
D	99	18	123	62	19	118
E	156	38	225	-	-	223
F	16	4	170	-	-	172
G	911	162	64	977	191	104
J	3821	1225	482	4330	1323	335
P	240	19	19	305	30	35
Q	4	1	1	4	1	1
X	-	-	8	40	4	12

Table 1: distribution of FATDS message traffic

partition	destination		source	
	device	tuples	device	tuples
0	A	1278	A	1142
1	C	3748	C	3405
2	D	240	D	199
3	E	419	E	223
4	F	190	F	172
5	G	1137	G	1272
6	J	5528	J	5988
7	P,Q,X	292	P,Q,X	432
mean	-	1604	-	1604

Table 2: sample partition for MESSAGE relation

5 Conclusions

1. The relative partition model can describe several types of data skew: tuple population skew, selectivity skew, hash partition skew, and join probability skew.
2. Given a relation and a partitioning scheme, the value of the skew parameter Q for each type of skew can be unambiguously calculated.
3. Under reasonable worse case assumptions, the existence of considerable data skew can be demonstrated in real databases.
4. Measurements of the two databases presented here support previous analytic work [26], which assumed Q values in the range 2-3 for TPS, SS, and HPS.

6 Acknowledgments

Lee Lauderadale of the College of Business Administration computer center provided invaluable assistance in obtaining and processing the stock trading data. The authors also thank Alfred Dale for valuable discussions and for reading several preliminary drafts of this report.

References

[1] Chaitanya K. Baru and Ophir Frieder. Database operations in a cube-connected multicomputer system. *IEEE Transactions on Computers*, C-38(6):920–927, June 1989.

[2] Chaitanya K. Baru, Ophir Frieder, Dilip Dandlur, and Mark Segal. Join on a cube: analysis, simulation, and implementation. In M. Ketsuregawa and H. Tanaka, editors, *Database Machines and Knowledge Base Machines*, pages 61–74, Kluwer Academic Publishers, Norwell, MA, October 1987. [Karuizawa, Nagano, Japan].

[3] Haran Boral. Parallelism and data management. In *Third International Conference on Data and Knowledge Bases*, June 1988. [Jerusalem, Israel].

[4] Stavros Christodoulakis. Estimating record selectivities. *Information Systems*, 8(2):105–115, 1983.

[5] Alfred G. Dale, F. Furman Haddix, Roy M. Jenevein, and Christopher B. Walton. *Scalability of Parallel Joins on High Performance Multicomputers.* Technical Report TR-89-17, University of Texas at Austin Department of Computer Sciences, Austin, TX, USA, June 1989.

[6] David J. DeWitt et al. Gamma - a high performance backend database machine. In *Twelth Very Large Data Base Conference*, 1986. [Kyoto, Japan, August 1986].

[7] R. A. Fairthorne. Empirical hyperbolic distributions for bibliometric description and prediction. *Journal of Documentation*, 25(4):319–343, 1969.

[8] Robert H. Gerber. *Dataflow Query Processing using Multiprocessor Hash-Partitioned Algorithms.* Technical Report 672, University of Wisconsin Computer Sciences Department, Madison, WI, USA, October 1986.

[9] Robert H. Gerber and David J. DeWitt. *The Impact of Hardware and Software Alternatives on the Performance of the GAMMA Database Machine.* Technical Report 708, University of Wisconsin Computer Sciences Department, Madison, WI, USA, July 1987.

[10] R.-C. Hu and R. Muntz. *Removing Skew Effect in Join Operations on Parallel Processors.* Technical Report CSD-890027, University of California at Los Angeles Computer Science Department, Los Angeles, CA, June 1989.

[11] Balakrishna R. Iyer, Gary R. Ricard, and Peter J. Varman. *An Efficient Percentile Partitioning Algorithm for Parallel Sorting.* Research Report RJ 6954, IBM Almaden Research Center, San Jose, CA, USA, August 1989.

[12] M. Kitsuregawa, M. Nakano, and T. Moto-Oka. Application of hash to database machine and its architecture. *New Generation Computing*, 1(1), 1983.

[13] M. Seetha Lakshimi and Philip S. Yu. Effect of skew on join performances in parallel architecture. In *Symposium on Databases in Parallel and Distributed Systems*, pages 107–120, August 1988. [Austin, TX, USA].

[14] M. Seetha Lakshimi and Philip S. Yu. Limiting factors of join performance on parallel processors. In *Conference on Data Engineering*, pages 488–496, February 1989. [Los Angeles, CA, USA].

[15] Lee Lauderdale, database adminstrator, College of Business Adminstration, University of Texas at Austin. private discussion, 25 January 1990.

[16] Raymond A. Lorie and Honesty C. Young. *A Low Communication Sort Algorithm for a Parallel Database Machine.* Research Report RJ 6669, IBM Almaden Research Center, San Jose, CA, USA, February 1989.

[17] C. A. Lynch. Selectivity estimation and query optimization in large databases with highly skewed distributions of column values. In *14th International Conference on Very Large Databases*, 1988.

[18] Michael V. Mannino, Paicheng Chu, and Thomas Sager. Statistical profile estimation in database systems. *ACM Computing Surveys*, 20(3):191–221, September 1988.

[19] Edward Omiecinski and Eileen Tien Liu. *The Adaptive-Hash Join Algorithm for a Hypercube Multicomputer.* Technical Report GIT-ICS-89/48, School of Information and Computer Science Georgia Institute of Technology, Atlanta, GA, December 1989.

[20] G. Piatetsky-Shapiro and G. Connell. Accurate estimation of the number of tuples satisfying a condition. In *ACM SIGMOD Conference*, pages 256–276, June 1984. [Boston, MA, USA].

[21] James P. Richardson, Hongjun Lu, and Krishna Mikkilineni. Design and evaluation of parallel pipelined join algorithms. In *ACM SIGMOD Conference*, pages 399–409, May 1987. [San Francisco, CA, USA,].

[22] G. Scarrot. Will zipf join gauss? *New Scientist*, 62(898):402–404, 1962.

[23] Donovan A. Schneider and David J. DeWitt. *A Performance Evaluation of Four Parallel Join Algorithms in a Shared-Nothing Multiprocessor Environment.* Technical Report 836, University of Wisconsin Computer Sciences Department, Madison, WI, USA, April 1989.

[24] Michael Stonebraker. The case for shared nothing. *Database Engineering*, 9(1), March 1986.

[25] Peter J. Varman, Balakrishna R. Iyer, and Donald J. Haderle. *Parallel Merge on an Arbitrary Number of Processors.* Research Report RJ 6632, IBM Almaden Research Center, San Jose, CA, USA, December 1988.

[26] Christopher B. Walton. *Four Types of Data Skew and their Effect on Parallel Join Performance.* Technical Report TR-90-12, Department of Computer Sciences, University of Texas at Austin, May 1990.

[27] Christopher B. Walton. *Investigating Skew and Scalability in Parallel Joins.* Technical Report TR-89-39, Department of Computer Sciences, University of Texas at Austin, December 1989.

[28] Joel L. Wolf, Daniel M. Dias, and Philip S. Yu. *An Effective Algorithm for Parallelizing Hash Joins in the Presence of Data Skew.* Research Report RC 15510, IBM T.J. Watson Research Center, Yorktown Heights, NY, USA, February 1990.

[29] Joel L. Wolf, Daniel M. Dias, and Philip S. Yu. *An Effective Algorithm for Parallelizing Sort Merge Joins in the Presence of Data Skew.* Research Report RC 15138, IBM T.J. Watson Research Center, Yorktown Heights, NY, USA, November 1989.

[30] G. K. Zipf. *Human Behavior and the Principles of Least Effort.* Addison-Wesley, Cambridge, MA, USA, 1949.