Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computer Communications 27 (2004) 1166–1173
computer communications
www.elsevier.com/locate/comcom

A performance anomaly in clustered on-line transaction processing systems

Hong Cai[a,*], Hisao Kameda[b], Jie Li[b]

[a]Doctoral Program in Engineering, University of Tsukuba, 1-1-1 Tennodai, Tsukuba-shi, Ibaraki-ken, 305-8573, Japan
[b]Institute of Information Sciences and Electronics, University of Tsukuba, 1-1-1 Tennodai, Tsukuba-shi, Ibaraki-ken, 305-8573, Japan

Abstract

This paper presents a simulation study on complex On-Line Transaction Processing systems using TPC-C workload. The impacts on the response time of Inter-Connection Network (ICN) in the system are studied. From the simulation results, we find anomalous cases where with the ICN that employs CSMA/CD protocol, increasing the number of database servers degrades the performance of the whole system. The explanation of the structure underlying this anomalous phenomenon is given after having investigated the behavior of the components in the system.

Keywords: On-line transaction processing; Inter-connection network; Performance evaluation; TPC-C workload; Simulation

1. Introduction

An On-line Transaction Processing (OLTP) system is an information system that is optimized for data entry, update, validation, and deletion. It is used to support business and operational processes within an organization, such as banking, airlines, mail-order, supermarkets, and manufacturers, etc. OLTP applications are client/server applications that give on-line users direct access to databases. The OLTP applications process units of work, which are called transactions [1]. As to banking systems, a single transaction might request a bank balance; another might update that balance to reflect a deposit.

Most companies designed their OLTP systems in a cluster style. A cluster environment is a collection of physical machines that, in most cases, communicate with each other over some private networks. Each node in the cluster has its own processor (or processors in the case of clustered SMP machines) and memory, but all nodes can access to a common set of disks for storage. Clustered systems are used to improve performance by providing scalability and to improve the availability of important data as opposed to a single physical machine.

The Transaction Processing Performance Council (TPC) [2] is a non-profit organization founded to define transaction processing and database benchmarks and to disseminate objective and verifiable TPC performance data to the industry. TPC Benchmark A (TPC-A) [3], issued as its first OLTP benchmark in November, 1989 (Obsolete as of 6/6/1995), is an industry standard benchmark which is based on the debit–credit transaction processing benchmark. It is intended to reflect an OLTP application, but it does not reflect the entire range of OLTP requirements typically characterized by multiple transaction types of varying complexities. In July of 1992, TPC Benchmark C (TPC-C) [4] was approved. It is different from and more complex than TPC-A in its multiple transaction types, more complex databases, and overall execution structure. TPC-C simulates a complete environment where a population of terminal operators execute transactions against a database. It is centered around the principal activities (transactions) of an order-entry environment. It tests a broad cross-section of database functionality including inquiry, update, and queued mini-batch transactions.

There has already been a significant amount of research into modeling and analyzing OLTP systems and workloads. Kohno and Kameda [5] presented the effects on the response time of the number of the hardware components and of the Inter-Connection Network (ICN). Yu [6] presented a general analytic

* Corresponding author. Tel/fax: +81-29-853-5156.
E-mail addresses: caihong@osdp.is.tsukuba.ac.jp (H. Cai), kameda@osdp.is.tsukuba.ac.jp (H. Kameda), lijie@osdp.is.tsukuba.ac.jp (J. Li).

0140-3664/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2004.02.024

modeling approach, and a general methodology to analyze various coherent control schemes. Leutenegger and Diaas [7] presented a modeling study on TPC-C workload. Xi et al. [8] presented a modeling approach to predict buffer pool hit rate for IBM DB2 Universal Database (DB2/UDB) with TPC-C workload. Du et al. [9] investigated the feasibility of using queueing network models with the support of simulation to study the symmetric multipleprocessor architectural impacts on performance under commercial workloads such as TPC-C. Alameldeen et al. [10] developed a simulation methodology based on multiple simulation processes, which scaled workloads (including TPC-C workload) down (in both size and time) to allow simulations on a $2K PC. Barroso et al. [11] examined the design trade-offs that arise as more system functionality is integrated onto the processor chip and presented a detailed study of the performance impact of chip-level integration in the context of OLTP workloads. Ramirez et al. [12] provided a detailed study of profile-driven compiler optimizations to improve the code layout in commercial workloads with large instruction footprints.

In this paper, we perform a simulation study on clustered OLTP systems using TPC-C workload. For this purpose, we build a simulator using an object-oriented approach. The simulation results show anomalous cases where if the system has an ICN that employs CSMA/CD protocol, increasing the number of database servers sometimes degrades the performance of the whole system. Kohno and Kameda [5] have also found a similar phenomenon. In their study, they used TPC-A as the workload. They also have found that the delay in the ICN increases rapidly as the number of servers increases and have stated that it is the cause of the phenomenon. However, they did not give further explanation on the reason why the ICN delay will increase. After having investigated the behavior of the components in the system, we are able to give a more reasonable explanation to this phenomenon.

The rest of the paper is organized as follows. In Section 2, we give the description of TPC-C workload, and then we describe a simulation model of clustered OLTP systems, as well as the simulator. Simulation results and discussions are given in Section 3. Conclusion appears in Section 4.

2. Workload, system model, and simulator

2.1. Workload

TPC-C is comprised of a set of basic operations designed to exercise system functionalities in a manner representative of complex OLTP application environments. The current version of the TPC-C is Version 5.

In the TPC-C business model, a wholesale parts supplier (called the Company below) operates out of a number of warehouses and their associated sales districts. The TPC benchmark is designed to scale just as the Company expands and new warehouses are created. However, certain consistent requirements must be maintained as the benchmark is scaled. Each warehouse in the TPC-C model must supply ten sales districts, and each district serves 3000 customers. Fig. 1 illustrates the warehouse, district, and customer hierarchy of the business environment of TPC-C.

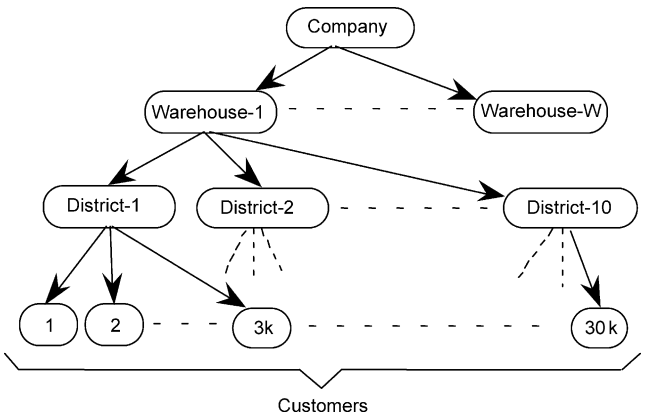

Fig. 1. The hierarchy of the business environment of TPC-C.

All warehouses maintain stocks for the 100,000 items to be sold by the Company. Customers call the Company to place a new order or request the status of an existing order. Orders are composed of an average of 10 order lines (i.e. line items). The system of the Company is also used to enter payments from customers, process orders for delivery, and to examine stock levels to identify potential supply shortages. The components of the TPC-C database are defined to consist of nine separate and individual tables: Warehouse, District, Customer, Order, New-Order, Order-Line, Stock, Item, and History. The relationships among these tables are defined in the entity–relationship diagram shown in Fig. 2.

- All numbers shown illustrate the database population requirements.
- The numbers in the entity blocks represent the cardinality of the tables (number of rows). These numbers are

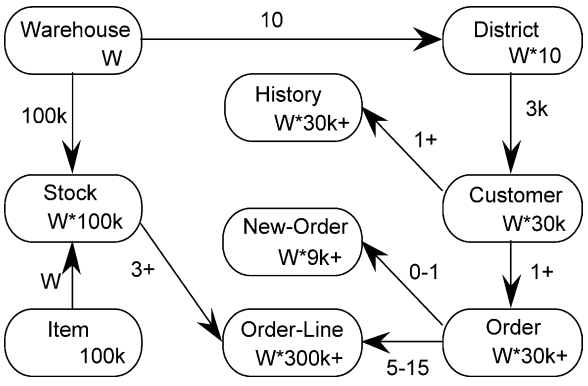

Fig. 2. Relationships among TPC-C tables.

factored by W, the number of Warehouses, to illustrate the database scaling.
- The numbers next to the relationship arrows represent the cardinality of the relationships (average number of children per parent).
- The plus (+) symbol is used after the cardinality of a relationship or table to illustrate that this number is subject to small variations in the initial database population over the measurement interval as rows are added or deleted.

Five types of transactions are defined in TPC-C, *New-Order*, *Payment*, *Order-Status*, *Delivery*, and *Stock-Level*. They operate on the tables mentioned above. An operator from a sales district can select, at any time, one of the five operations or transactions offered by the order-entry system of the Company. Like the transactions themselves, the frequency of the individual transactions are modeled after realistic scenarios. The *New-Order* transaction places an order (of 10 order lines) to a warehouse through a single database transaction. It inserts the order and updates the corresponding stock level for each item. It is the backbone of the workload. The *Payment* transaction processes a payment for a customer, updates the customer's balance, and reflects the payment in the district and warehouse sales statistics. The *Order-Status* transaction returns the status of a customer's last order. The *Delivery* transaction processes orders corresponding to 10 pending orders, one for each district, with 10 items per order. The *Stock-Level* transaction examines the quantity of stock for the items ordered by each of the last orders in a district and determines the items that have a stock level below a specified threshold.

The performance metric for TPC-C is expressed in transactions-per-minute-C (tpm-C); the tpm-C metric is the number of *New-Order* transactions executed per minute. The throughput of the TPC-C is driven by the activity of the terminals connected to each warehouse. For each active warehouse in the database, 10 terminals are associated. To increase the throughput, a larger number of warehouses and their associated terminals must be configured.

2.2. TPC-C configuration

A typical TPC-C configuration [13] is shown in Fig. 3. Transactions are generated by emulated users. An emulated user selects a transaction type, inputs a transaction of that type (keying), waits for the output of the transaction, and thinks after getting the output on the screen.

Transaction selection occurs at random but is based on a minimum mixed percentage for each transaction type. The Keying Time is constant and must be a minimum of 18 s for New-Order, 3 s for Payment, and 2 s each for Order-Status, Delivery, and Stock-Level. For each transaction type, the minimum mixed percentage and the keying time are listed in Table 1.

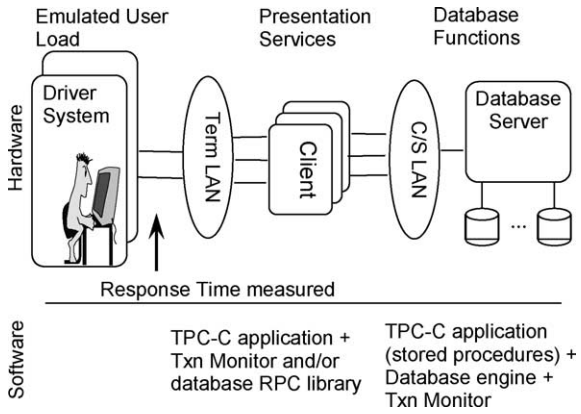

Fig. 3. Typical TPC-C configuration.

2.3. The queueing model of the system

Responding to the demand of higher performance systems in the market, more and more large-scale commercial products have appeared, such as HP ProLiant series [14], Fujitsu PRIMEPOWER series [15], IBM @Server xSeries [16], etc. These computer manufacturers also have issued some TPC-C performance results that can be found on the Internet [17].

Most of the products employed similar Client/Server(C/S) architectures, with little variation. The database managers installed on the servers could be Oracle database, Microsoft SQL Server, IBM DB2/UDB, etc. In this study, referring to the systems mentioned above, we take such an architecture with some simplifications as the simulation target. The queueing model of the system is depicted in Fig. 4.

The followings are the components of the system.

- Terminal—transactions are generated and finished here.
- AP_U—Application Processor Unit, is a client, which is dedicated to executing OLTP transactions; contains Application Manager, Transaction Manager (TM).
- DBP_U—Database Processor Unit, is a server, which provides database-processing services; contains Resource Manager, Concurrent Control Manager, etc.
- ICN—Interconnection Network, is the C/S LAN.
- DK_U—Disk Unit, is the storage.

Table 1
The minimum percentage of mix, keying time for each transaction type

Transaction type	Minimum % of mix	Keying time (s)
New-Order[a]	n/a	18
Payment	43.0	3
Order-status	4.0	2
Delivery	4.0	2
Stock-level	4.0	2

[a] There is no minimum for the New-Order transaction as its measured rate is to be the reported throughput.

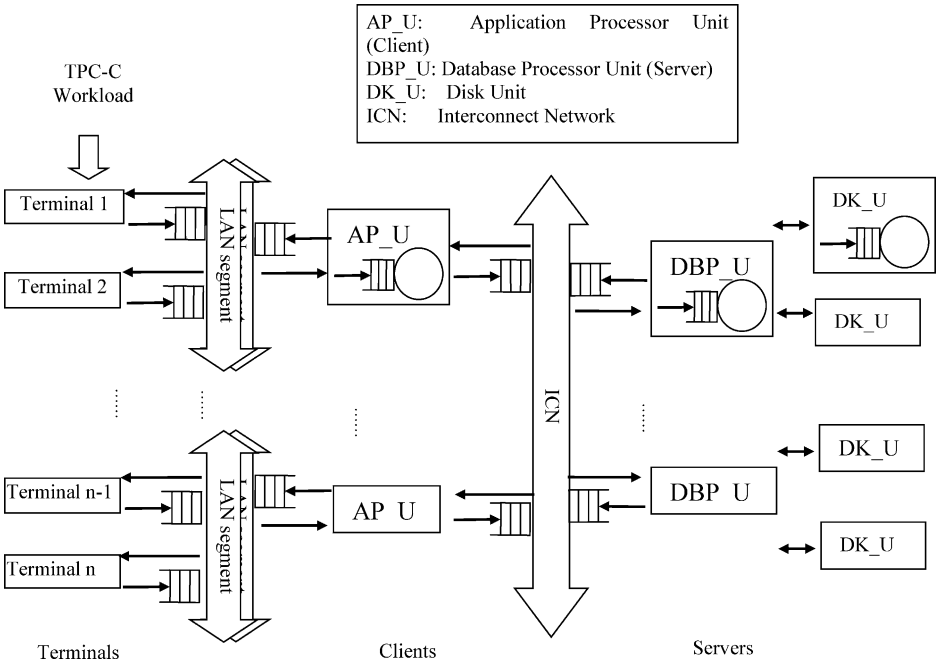

Fig. 4. The queueing model of the system.

Normally, one of the five types of TPC-C transactions is generated and enters the system from one of the terminals, which is connected to a dedicated client machine (AP_U) via a LAN segment. At AP_U, the transaction is processed according to its profile. This is controlled by Application Manager and TM. When database access occurs, the request is sent to the corresponding DBP_U through the ICN. DBP_U will handle the database operations under the Concurrency Control Manager. Then results will be sent back to the AP_U. Finally, when the transaction is committed successfully, results will return to the terminal where the transaction has been generated. After a random thinking time and keying time, a new transaction is generated and a similar round begins. If a transaction is aborted, it will be restarted after a delay time.

In our study, we choose Ethernet hubs of different speeds as the ICN, in which the CSMA/CD protocol is employed. For a comparison, we also study the system with the ICN of a FastEthernet Switch (Cisco 6500) [18].

2.4. Simulator

Our simulator is written in C++, with the tool of OOSim++ [19]. OOSim++ version 1.0 is a software toolkit for event scheduling and queueing model simulations, developed at the CISE Department, University of Florida, Gainesville FL, USA. It is written in C++ and adhering to object-oriented design. OOSim++ is rooted in the SimPack simulation package, a collection of C++ tools (routines and programs) for computer simulation. This is a set of routines that are functionally very similar to the routine library called *smpl* by MacDougall [20].

In the simulator, a virtual base class of XObject is defined. All the other classes in the simulator are derived from XObject. Each type of component (such as AP_U, DBP_U, ICN, etc.) and utility (such as Application Manager, Transaction Manager, Concurrent Control manger, Commit manager, etc.) is defined as an object class. Arrival and departure of a transaction, request and release of a CPU, and so on are all defined as events. A dispatcher is used to handle all these events and dispatch them to their appropriate handlers. The code is developed in modular style, and most of the parameters are adjustable at running time. It is easy to add new features and new components into our simulator.

2.5. Transaction profiles

The New-Order transaction consists of entering a complete order through a single database transaction. It represents a mid-weight, read-write transaction with a high frequency of execution and stringent response time requirements to satisfy on-line users. This transaction type is the backbone of the TPC-C workload. Taking the New-Order transaction as an example, we show the detailed database operations of a transaction as follows.

BEGIN TRANSACTION
  Select warehouse-id from WAREHOUSE table
  Select district-id, warehouse-id from DISTRICT table
  Update district-id, warehouse-id from DISTRICT table
  Select customer-id, district-id, and warehouse-id from CUSTOMER table

Insert into ORDER table
Insert into NEW-ORDER table
For each of 5-15 items
Select item-id from ITEM table
Select item-id, warehouse-id from STOCK table
Update item-id, warehouse-id in STOCK table
Insert into ORDER_LINE table
COMMIMIT TANSACTION

2.6. Assumptions

To simplify the configurations of the simulator, we assume that the overhead of deadlock detection can be ignored. We also assume that the mean service time for each step in the transaction profile executed in AP_Us is identical, so is in DBP_Us. When we need to access a record, we first check if it is in the main memory (buffer). If not, we read it from a disk with mean disk access time; else, we get it directly from the memory. From Ref. [8], we can assume that the buffer-pool size is large enough, and the buffer-hit ratio is constant. The concurrent control method [21,22] we apply to the simulator is the two-phase lock (2PL) method, in which three kinds of lock modes are supported: shared (read), exclude, and update (first shared, and then exclude). In the simulator, the deadlock is detected by means of Wait-for-Graph (WFG); the transaction that causes the deadlock is aborted and restarted after a random time.

3. Results and discussion

Since the TPC-C defines the number of New-Order transactions executed per minutes as the system performance metric, we show the simulation results of New-Order transaction here. We simulated the systems with the ICN of the Ethernet hubs of different speeds. First, we present the results for the case of the Ethernet hub of 10 Mbps, and then the results of using FastEthernet hub. In both cases, the CSMA/CD protocol is employed. Finally, we present the results of using FastEthernet switch for comparison.

Table 2
The parameters of the simulator (1)

The number of terminals	480
The number of AP_Us	8
The number of CPUs in AP_U	1
The number of DBP_Us	1–8
The number of CPUs in DBP_U	1
Mean service time in AP_Us	0.05 ms
Mean service time in DBP_Us	0.05 ms
Mean disk access time	5 ms
Buffer-hit ratio	95%
Confidence level	95%
Confidence interval	5%

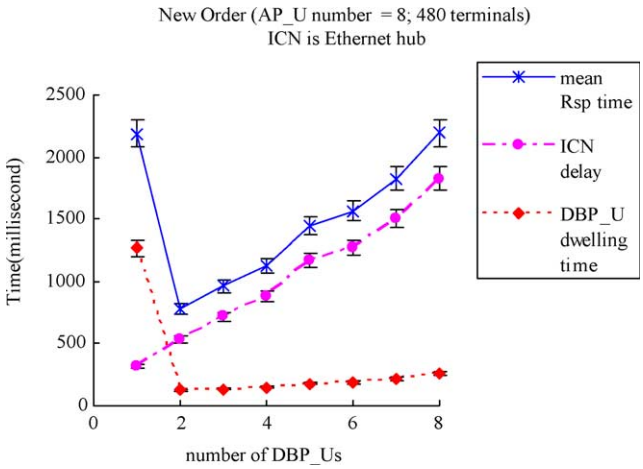

Fig. 5. The results of using 10 Mbps Ethernet hub.

3.1. The results of using 10 Mbps Ethernet hub

The simulation parameters are listed in Table 2 Fig. 5 shows the mean response time vs. the number of DBP_Us. We observe that the mean response time decreases as the number of DBP_Us changes from 1 to 2, and then increases. In the same figure, we also show the mean ICN delay and the mean dwelling time in the DBP_Us. We observe that as the number of DBP_Us increases, the ICN delay increases rapidly while the DBP_U dwelling time first decreases, and then increases.

3.2. The results of using 100 Mbps Ethernet hub

We show here the simulation result of the system with the ICN of the 100 Mbps FastEthernet hub The execution speed of the AP_Us and the DBP_Us are changed. Correspondingly, the number of the terminals is increased as well, which means heavier loads to the system. The parameters of the simulation are listed in Table 3 (The values of the parameters different from those listed in Table 2 are shown in italic).

Fig. 6 shows the mean response time vs. the number of DBP_Us. We observe that the mean response time decreases as the number of DBP_Us changes from 2 to 3, and then increases. Similarly, we show the mean ICN delay

Table 3
The parameters of the simulator (2)

The number of terminals	*4800*
The number of AP_Us	8
The number of CPUs in AP_U	1
The number of DBP_Us	2–16
The number of CPUs in DBP_U	1
Mean service time in AP_Us	*0.01 ms*
Mean service time in DBP_Us	*0.01 ms*
Mean disk access time	5 ms
Buffer-hit ratio	95%
Confidence level	95%
Confidence interval	5%

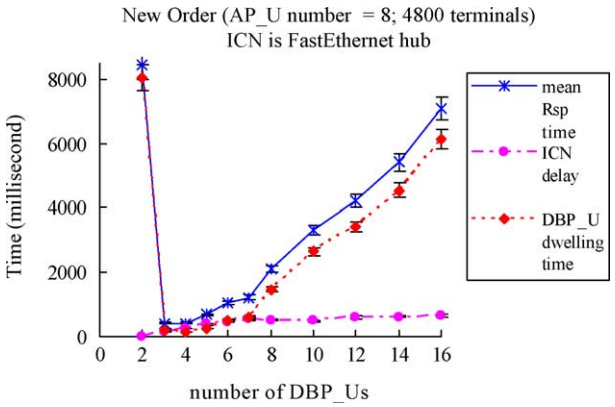

Fig. 6. The results of using 100 Mbps Ethernet hub (1).

and the mean dwelling time in the DBP_Us in the same figure. Differently from the Fig. 5, we observe that as the number of DBP_Us increases, the ICN delay increases at first, and then it becomes stable while the DBP_U dwelling time first decreases, and then increases rapidly.

To see into what happens in the DBP_Us, we show the mean DBP_U dwelling time and the mean transaction block latency time in Fig. 7. Transaction block latency time is the total time a transaction spent in DBP_Us waiting for requested locks to be granted. We found that the mean block latency time is the most part of the mean DBP_U dwelling time.

3.3. The results of using 100 Mbps FastEthernet switch

For a comparison, we replace the ICN of the above system with that of a FastEthernet switch The main difference is that the switch does not use the CSMA/CD protocol. There is a high-speed bus backplane, which can pass packets from the source port to the destined port very fast.

The parameters of the simulation are the same as in Table 3. Fig. 8 shows the effect of the number of DBP_Us on the mean response time. We observe that as the number of DBP_Us increases, the system performance goes steady

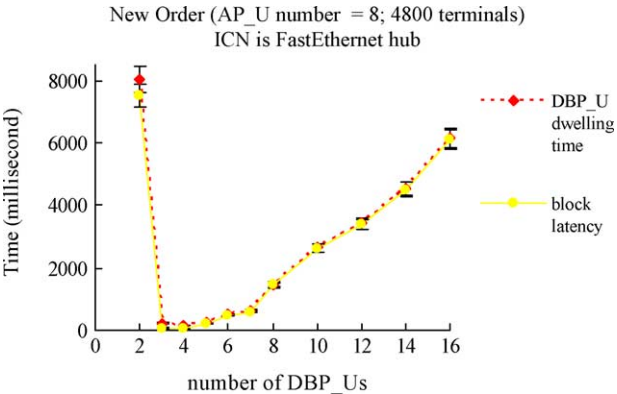

Fig. 7. The results of using 100 Mbps Ethernet hub (2).

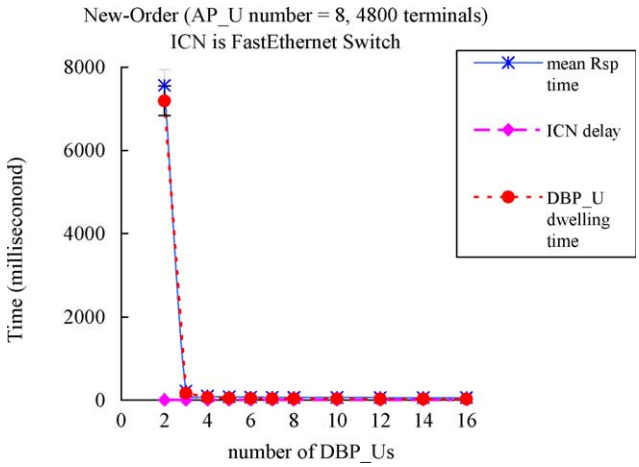

Fig. 8. The results of using 100 Mbps FastEthernet switch.

after 4. We cannot observe obvious degradation in the system performance.

3.4. Discussion

Figs. 3 and 4 show a trend that increasing the number of DBP_Us degrades the system performance. Kohno and Kameda [5] have also found a phenomenon similar to that shown in Fig. 5. In their study, they found that the system performance would degrade as the number of DBP_Us increases. They stated that the reason was that the ICN delay increased rapidly as the number of DBP_Us increased. However, they did not give explanation on the reason why the ICN delay increased as the number of DBP_Us increased. In this paper, after having investigated the behavior of the components in the system, we give an explanation to this phenomenon from the behavior of both the TM and the ICN in the following.

The role of the TM in the system is coordinating the initiation and the completion of transactions among the DBP_Us. It performs the transaction commit, rollback, and recovery. One of its primary functions is to enforce the two-phase commit (2PC) protocol, which ensures an atomic commit. According to this, TM requires that

- at the beginning of the transaction, all the DBP_Us should be notified of Transaction Start;
- at the ending of the transaction, all the DBP_Us should be notified of Transaction Prepare-commit, and then Transaction Commit or Aborted (if it cannot successfully commit);
- the transaction will not proceed until all of the DBP_Us have replied positively in previous two cases.

These communications take place among the AP_Us and the DBP_Us, which means that they consume some bandwidth of the ICN. Therefore, as the number of DBP_Us increases, the number of the packets over the ICN will increase.

For the sake of clarity, we do some simple calculations here. Taking a New-Order transaction as example, during its lifetime, its total number of packets passing through the ICN is

$$N_{\text{total}} = N_{\text{tm}} + N_{\text{appm}},$$

where N_{tm} represents the number of packets related to TM operations and N_{appm} represents the ones related to Application Manage operations. Here the case of abortion and restart of the transaction is not taken into consideration.

Regarding the TM, there are three notification packets: Transaction Start, Transaction Pre-commit, and Transaction Commit, which are broadcast to all the DBP_Us. The number of reply packets will be $3N$, where N is the number of DBP_Us, so

$$N_{\text{tm}} = 3 + 3N.$$

Regarding the Application Manager, every time when there occurs a database operation (such as *select*, *update*), a packet of request will be sent to a server (DBP_U), and a packet of reply will come back from the server. The profile of the New-Order transaction starts with six steps, followed with a loop containing four steps. The loop iterates in average 10 times. Each step generates a packet of request and a packet of reply. Thus, the average N_{appm} for a New-Order transaction is

$$N_{\text{appm}} = (6 + 4 \times 10) \times 2.$$

Then the average N_{total} for a New-Order transaction can be obtained as

$$N_{\text{total}} = 3N + 95.$$

Table 4 shows the calculation results of 1–8 DBP_Us.

Upon the above calculations, we know that when the number of DBP_Us changes from 1 to 8, the average number of packets over the ICN for a New-Order transaction will increase to about 21%, which means the loads on the ICN will get heavier.

When the ICN is a 10 Mbps Ethernet hub that employs the CSMA/CD protocol, we know that heavy loads will cause a lot of collisions and retries in the ICN, which will cause significant congestion and delay. Therefore, the ICN delay will increase hastily as the number of DBP_Us increases. This can explain Fig. 5 and the results in the Kohno and Kameda's study [5].

Table 4
The average total number of ICN packets for a New-Order transaction

The number of DBP_Us	Average N_{total} (New-Order)	Increment (%)
1	98	–
2	101	3.06
3	104	6.12
4	107	9.18
5	110	12.2
6	113	15.3
7	116	18.4
8	119	21.4

On the other hand, when the system has the ICN with the 100 Mbps FastEthernet hub and more terminals, much more transactions are generated in the system. As the number of DBP_Us increases, the ICN delay will increase at first. Then, the increase in the ICN delay makes the transactions stay longer in the system, and in turn, causes more data contentions and blocks in the DBP_Us. Thus, more transactions are likely to be blocked in the DBP_Us (as shown in Fig. 7). That means that transactions will spend much more time in the DBP_Us. Thus, the loads on the ICN are alleviated, and the ICN delay becomes stable. That is what happens in Fig. 6.

In Figs. 5 and 6, although the phenomenon in the mean response time is similar, there are some differences. In Fig. 5, the ICN speed and the system loads are comparatively low. When the number of DBP_Us increases, the ICN delay increases rapidly and it is the main part in the system mean response time. In Fig. 6, the ICN speed is faster and the system loads are heavier. When the number of DBP_Us increases, firstly, the ICN delay increases, then because of more blocks in the DBP_Us, more transactions are waiting in the DBP_Us. In the latter case, the loads on the ICN are alleviated and the ICN delay becomes stable after some point, while the dwelling time in the DBP_Us will increase rapidly.

In contrast, we did not observe similar results in the case of using a FastEthernet switch in Fig. 8. The ICN delay caused by increasing the number of DBP_Us is negligible.

In summary, our explanation is that, since the TM broadcasts the notification of the starting, the pre-committing, and the committing of a transaction to all the DBP_Us and waits for their replies, the increase in the number of DBP_Us will lead to the increase in the loads over the ICN. If we choose an ICN that employs the CSMA/CD protocol, the performance of the ICN will degrade because of increased collisions and retries. Moreover, the transactions will stay longer in the system and are likely to cause more blocks in the DBP_Us. Therefore, the performance of the whole system will degrade.

4. Conclusion

We have studied the role of the ICN in a clustered OLTP system with simulation. We found an anomalous phenomenon that in the system with the ICN of an Ethernet hub that employs the CSMA/CD protocol, as the number of DBP_Us increases, the system performance degrades. We have discussed the internal structure underlying this anomalous phenomenon.

References

[1] J. Gray, Online Transaction Procesing Systems, McGraw-Hill, 1994.
[2] http://tpc.org.
[3] http://www.tpc.org/tpca/
[4] http://www.tpc.org/tpcc/
[5] K. Kohno, H. Kameda, Modelling and simulation of parallel on-line transaction processing systems, International Journal of Modelling and Simulation 19 (2) (1999) 158–164.
[6] P.S. Yu, Modeling and analysis of transaction processing systems, Lecture Notes in Computer Science 729 (1993) 651–675.
[7] S. Leutenegger, D. Diaas, A Modeling Study of the TPC-C Benchmark, ACM SIGMODS, 1993, pp. 22–31.
[8] Y. Xi, P. Martin, W. Powley, An analytical model for buffer hit rate prediction, Proc. of CASCON 2001, Toronto, 2001.
[9] X. Du, X. Zhang, Y. Dong, L. Zhang, Architecture effects of symmetric multiprocessors on TPC-C commercial workload, Journal of Parallel and Distributed Computing 61 (2001) 609–640.
[10] A.R. Alameldeen, M.M.K. Martin, C.J. Mauer, K.E. Moore, M. Xu, D.J. Sorin, M.D. Hill, D.A. Wood, Simulating a $2M commercial server on a $2K PC, IEEE Computer 36 (2) (2003) 50–57.
[11] L.A. Barroso, K. Gharachorloo, A. Nowatzyk, B. Verghese, Impact of Chip-Level Integration on performance of OLTP workloads, Proceedings of the Sixth International Symposium on High-Performance Computer Architecture, Toulouse, France, 2000, pp. 3–14.
[12] A. Ramirez, L.A. Barroso, K. Gharachorloo, R. Cohn, J. Larriba-Pey, P.G. Lowney, M. Valero, Code layout optimizations for transaction processing workloads, Proceedings of the 28th annual international symposium on Computer architecture, Göteborg, Sweden, 2001, pp. 155–164.
[13] SIGMOD'97 Industrial Session 5, Standard Benchmarks for Database Systems. http://www.tpc.org/information/sessions/sigmod/sigmod97.ppt.
[14] HP ProLiant DL servers, http://h18004.www1.hp.com/products/servers/platforms/index-dl-ml.html
[15] Fujitsu PRIMEPOWER, http://primeserver.fujitsu.com/primepower/
[16] IBM @Server xSeries, http://www.pc.ibm.com/us/eserver/xseries.
[17] TPC-C Benchmark Results by Performance, http://tpc.org/tpcc/results/tpcc_perf_results.asp.
[18] Cisco Catalyst 6500 White Paper, http://www.cisco.com/warp/public/cc/pd/si/casi/ca6000/tech/c65sp_wp.htm
[19] R. M. Robert, P. A. Fishwick, Sim++ Reference Manual, http://www.cis.ufl.edu/~fishwick/simpack/simpack.html.
[20] M.H. MacDougall, Simulating Computer Systems: Technique and Tools, MIT Press, Cambridge, MA, 1987.
[21] P.A. Bernstein, N. Goodman:, Concurrency control in distributed database systems, ACM Computing Surveys 13 (2) (1981) 185–221.
[22] A. Thomasian, Concurrency control: methods, performance, and analysis, ACM Computing Surveys (CSUR) 30 (1) (1998) 70–119.