Performance Analysis of Concurrency Control Mechanisms for OLTP Databases

Samuel Kaspi and Sitalakshmi Venkatraman, *Senior Member, IACSIT*

***Abstract*—Concurrency control (CC) in distributed and multidimensional databases is becoming more important due to recent increase in high-volume data storage with increasing online transaction processing (OLTP) requirements for medium and large organisations. This paper examines three concurrency control mechanisms commonly adopted and analyses their performance in distributed databases for OLTP operational systems of enterprises. The three CC mechanisms investigated are, two phase locking (2PL), wait depth limited (WDL) and optimistic concurrency control. These CC mechanisms have been studied well in disk-based systems. However, with the recent advances of cost-effective main memory or in-memory storage that can support much higher transaction rates than disk-based systems, there is sufficient motivation to re-investigate the performance of such CC mechanisms in diverse processor configurations. This paper presents a comparison of their behaviour and performance in terms of throughput rates achieved with varying transaction size and contention. The outcome of this study has resulted in further research proposals for improving the performance of these CC mechanisms for OLTP databases.**

***Index Terms*—Concurrency control, online transaction processing (OLTP), in-memory databases, performance.**

I. Introduction

In today's enterprise computing environment, distributed and multidimensional data manipulation, processing and analysis play a vital component in a wide range of business applications. With recent developments to address scalability, bandwidth, privacy and multiple data owner issues, cloud computing and other distributed environments provide the need to investigate distributed data analysis methods and online transaction processing (OLTP) requirements for providing a highly reliable and real-time system [1]-[3]. Here, the main issue when attempting to manage and control distributed database systems is concurrent execution of transactions [4], [5]. Unlike centralised databases, distributed databases have additional factors such as data fragmentation, partitioning and replication to be considered [6], [7]. Conflicts may arise in transactions that lead to concurrent read and write operations on the same data item of multidimensional or distributed databases [8]-[10]. Hence, concurrency control (CC) in distributed databases becomes important to coordinate and serialise various concurrently executed transactions that require shared data access, without interfering with one another.

Manuscript received September 15, 2013; revised December 9, 2013.

The authors are with the Department of Higher Education (IT) - Business, Northern Melbourne Institute of TAFE, VIC 3066 Australia (e-mail: SamKaspi@nmit.edu.au, SitaVenkat@nmit.edu.au).

Concurrency control mechanisms have been researched for several years in the past with centralised databases [11]-[13]. However, the recent increased use of distributed databases in a wide range of business applications calls for their attention in database research [14]-[16]. In traditional distributed databases, each global transaction ensures that each sub-transaction runs at one of the nodes of the system with local data access. Hence, CC mechanisms should cater to additional complexity of information processing at every local level in the distributed databases apart from ensuring concurrency at global level of the system. In this paper, a review of CC mechanisms is conducted and their performance in distributed database environments that require high volume online transaction processing and data-intensive queries is investigated.

With commodity servers offering terabytes of main memory and the continuing decline of memory prices, it becomes cost-effective to have majority of distributed and mutidimensional database transactions or OLTP databases to fit entirely within main memory or in-memory [2], [9], [17]. The largest working datasets for OLTP could be accommodated in memory, and only infrequent transactions could refer to external storage [18]. In addition, this would reduce the overheads and issues of disk-based distributed CC. Since in-memory storage provides a much higher transaction rate as compared to disk-based transactions, this study focuses on investigating the performance of CC mechanisms for in-memory databases.

The rest of the paper is organized as follows. Section II describes the three CC mechanisms under study, namely two phase locking (2PL), wait depth limited (WDL) and optimistic CC. The experimental simulation conducted and performance results obtained are presented in Section III. Section IV provides an analysis and finally, Section V offers concluding remarks and future work.

II. Concurrency Control Mechanisms

A. Two Phase Locking (2PL)

The 2PL refers to the standard concurrency control policy where locks are applied by a transaction to data and released in two phases so that any conflict arising due to other transactions accessing the same data is addressed. In the first phase, 2PL allows for say *n* transactions into the system, acquiring locks for all its data objects, thereby blocking access to them. Here, mutual blocking between transactions results in a deadlock, which requires to be resolved by aborting a transaction or other means to release the lock and executing the transaction again. In the second phase, locks of completed transactions get released; thereby blocked

transactions get woken up acquiring locks from the point at which they were blocked. Apart from hardware considerations, the throughput of 2PL systems depends on how many transactions are successful in obtaining all their locks. This in turn, depends on the probability of conflict between transactions – or, as it is commonly known, contention and the length of time that transactions are allowed to remain blocked before they are timed out (that is, the cyclical behaviour of transactions). The main factors governing contention are –

1) The size of the database – the smaller the database, the greater the likelihood of conflict between transactions requesting objects from this database.
2) The number of objects required by transactions – the greater the number of objects required by transactions, the greater the probability of conflict between transactions.
3) The number of transactions in the system (concurrency) either processing or blocked – for any level of contention above 0, the higher the concurrency, the greater the probability of conflict that eventually leads to a decrease in throughput.

According to literature studies conducted [19], [20], assuming there are uniform transactions having equal probability with n as the concurrency level and p as the level of contention, the predicted transaction throughput is approximated by the following formula:

$$n\,(1\text{-}p/2)^{(n\text{-}1)}$$

The cyclical behaviour of transactions in 2PL systems implies that unless some preventive action is taken, throughput will decrease every cycle since the proportion of transactions in the wait chain grows. We propose that for any equivalent total processing capacity, a greater throughput is achieved by increasing the number of cycles that can be completed in a given time than by increasing the number of concurrent transactions processing in the same period of time. Theoretically, our proposition holds for all levels of contention greater than 0 but becomes more pronounced as contention increases.

In a distributed environment, 2PL concurrency control faces additional issues. When data are partitioned, automatic global deadlock resolution requires intensive research attention, and where data are not partitioned among atomic commitment protocol participants, distributed deadlocks need to be resolved with sophisticated and extended 2PL CC mechanisms. Since 2PL systems are susceptible to thrashing (degradation in system performance) in later cycles, CC methods that eliminate or reduce wait chains could increase throughput of transactions. We examine two well-known alternatives to 2PL next, namely, optimistic concurrency control and wait depth limited (WDL) by eliminating or reducing wait chains.

B. Optimistic Concurrency Control

Optimistic CC mechanism assumes that multiple transactions can complete without affecting each other and does not lock data objects until committing a transaction, when it verifies that no other transaction has modified its data. When conflicting modifications are identified then rollback occurs. There are two features of this system that help to improve its performance relative to 2PL. Firstly, because transactions that conflict with committing transactions are aborted and then restarted, there is no chain of transactions waiting behind locked transactions. Secondly is the fact that to be aborted, a transaction must not only conflict with another transaction, the transaction with which it conflicts must be a committed transaction. This property is known as essential blocking and the number of successful transactions is unbounded and increases logarithmically with concurrency [21], [22]. In [19] an upper and lower bound is provided for the number of successful transactions at each concurrency level. Where p is the level of contention and n the level of concurrency, these are defined by the two equations. The lower bound is defined as –

$$1 + (1 - p)/p + ln(p(n - 1) + 1)$$

while the upper bound is defined as –

$$1 + 1/p + ln(p(n - 1) + 1)$$

An interesting consequence is that while they allow higher throughputs at higher concurrencies than does 2PL, they imply that as for 2PL, for any equivalent total processing capacity, a greater throughput is achieved by increasing the number of cycles that can be completed in a given time than by increasing the number of concurrent transactions processing in the same period of time. In other words, for any level of contention and concurrency, throughput will be the same in every cycle. Thus, in an in-memory system, doubling processor speed doubles the number of cycles completed in a given time period and thus doubles throughput. However, doubling concurrency only increases throughput logarithmically. This is based on the assumption optimistic systems in identical hardware capability will have the same throughput with same values of p and n. In reality, both variability in transactions size and contention have a significant impact on system performance. An alternative CC mechanism commonly used is the Wait Depth Limited (WDL), which is examined next.

C. Wait Depth Limited (WDL)

A class of novel set of algorithms based on running priority/wait depth limited (WDL) attempt to minimize the length of wait chains by aborting non-active transactions that are blocking other transactions [23]. Under WDL, longer transactions are given priority over shorter transactions, and, herein lays the major difference between WDL with a wait depth of 1 and running priority. This preference given to larger transactions reduces the quadratic effect that is, the tendency of longer transactions to suffer a disproportional rate of restarts relative to shorter transactions. Thus, under WDL, larger transactions do not suffer the disadvantages that they face in optimistic and running priority systems. This tends to reduce the problem of wasted work since large transactions have in general performed more work when aborted than have smaller transactions. As well, when

compared to optimistic systems, since not all blocked transactions are restarted, one would expect less wasted work under both running priority and WDL than under optimistic systems. The cost of wasted work is largely dependant on the speed at which the work can be done. An important factor in determining speed is the speed of the processor. The cost of wasted work is largely dependant on the speed at which the work can be done. An important factor in determining speed is the speed of the processor.

III. Experimental Design

The experimental design of the system is composed of the following subsystems - a concurrency control subsystem, a database subsystem, a hardware subsystem a processing subsystem and a transaction subsystem. The specification of these subsystems are described below.

A. Database and Transaction Subsystems

There are two object stores available to transactions from which to choose their objects:- D1, which contains 1000 objects, and D2, which contains a million objects. Characteristics of objects in these data stores are:

1) In each data store each object is unique.
2) Data stores are disjoint.
3) Given a data store *D*, any two objects *Oi* and *Oj* in that data store have an equal probability of being required in any access. Thus for *D*1 which contains 1000 objects, each object in *D*1 has a 1/1000 chance of being required in an access to *D*1. Similarly, any object in *D*2, which contains a million objects, has a 1/1000000 chance of being required in an access to *D*2. Each of our systems System contains four transaction types or classes - *T*1, *T*2, *T*3 and *T*4. These transaction types require 1, 2, 4 and 8 objects respectively from *D*1 and 3, 6, 12 and 24 objects respectively from *D*2. *T*1 transactions represent 20% of transactions *T*2 transactions represent 20% of transactions , *T*3 transactions represent 35% of transactions and *T*4 transactions represent 25% of transactions.

B. Hardware Subsystems

OLTP applications minimise network overheads and lock contentions since they reduce round-trips between client and databases in a distributed environment. OLTP in-memory databases could run on hardware subsystems consisting of a cluster of shared-nothing main memory-only nodes [2], [5]. Their salient features are that the transactions are short-lived, repetitive and require access to only a small subset of data objects. Hence, the in-memory subsystems with processing power of 100 Mega instructions per second (MIPS), 200 MIPS, 1 Giga instructions per second (GIPS) and 2 GIPS are used for performance evaluation of the three CC mechanisms under the following four different hardware subsystems:

1) A subsystem containing 96 processors each operating at 100 MIPS.
2) A subsystem containing 96 processors each operating at 200 MIPS.
3) A subsystem containing 20 processors each operating at 1 GIPS.
4) A subsystem containing 10 processors each operating at 2 GIPS.

Besides measuring the relative performance of the three CC schemes, these four subsystems allow us to compare the performance of systems with fewer and faster processors as against those with more numerous but slower processors. In other words, with 96 processors, 20 processors and 10 processors, we try to double the processing speed to see the effect in throughput.

C. Concurrency Control Subsystems

Three basic concurrency control subsystems are used in this chapter these are 2PL, wait depth limited (WDL) and optimistic.

The 2PL subsystem is a standard 2PL system (including a deadlock breaking mechanism) except that for simplification, all requests for locks are requests for a write and the granularity of lock acquisition and release is a single lock.

The WDL subsystem is as described in Section 2C with a depth of 1 and the length cost as described in [17]. As with the 2PL system, all requests for locks are requests for a write and the granularity of lock acquisition and release is a single lock.

For testing the performance of Optimistic CC systems, we use a pure kill system since this minimizes the amount of wasted work by killing a transaction immediately, and assumes no access invariance. As with the previous systems, all objects are acquired for a write and the granularity of object acquisition is a single object.

D. Processing Subsystem

The processing system used for the experimental study has three stages, namely initialization, processing and completion. The initialization stage, whether for new or restarted transactions requires 100000 instructions per transaction regardless of the transaction's size. This stage requires CPU processing only. Similarly, the completion phase requires 50000 CPU instructions per committing transaction. The four hardware subsystems described above are tested for one second at each concurrency level of the three CC mechanisms. All the results shown are based on the average taken over 40 simulation runs conducted at each concurrency level for each system.

IV. Performance Results

In this section we present and analyze the results of the tests outlined in the previous section to compare the performance of concurrency control mechanisms for in-memory OLTP databases. Given that under any concurrency management system, the *proportion* of successful transactions is determined solely by the level of contention and the level of concurrency, one would expect that for any level of concurrency and contention throughput should scale up in linear proportion to processor speed. For example, in an in-memory system, given a concurrency management scheme at a concurrency of 100, and a given contention and processor speeds of 100 megahertz, with a throughput of 1000 transactions per second, one would expect that by maintaining all other parameters but doubling the processor speed would also double throughput. In fact our simulations indicate that this is the case. Table I provides the

results of comparing the ratios of throughput for simulations performed for in-memory 2PL, Optimistic and WDL CC mechanisms.

It should be noted that at the higher processor speeds, we only ran our simulations at lower concurrencies (5, 10, 15, 20 for 1 GIPS and 5, 10 for 2 GIPS). This is because the large number of transactions that needed to be generated at the higher processor speeds made the simulations prohibitive with the available resources. However, given the constant relationships indicated above and implied by the contention equations, the data for the higher contentions can be safely constructed by simply scaling the data from the lower processor speeds.

TABLE I: COMPARISON OF THROUGHPUT RATIOS WITH PROCESSOR SPEEDS

Processor Speed	2PL	Optimistic CC	WDL
100 MIPS vs 200 MIPS	1.941734687	2.008265414	1.99466704
100 MIPS vs 1 GIPS	9.561359765	9.74772916	9.90655507
100 MIPS vs 2 GIPS	19.03805469	19.59558478	19.9423341
200 MIPS vs 1 GIPS	4.915351509	4.891139981	4.9937712
200 MIPS vs 2 GIPS	9.70587829	9.952051527	10.0702565

A. Throughput with Varying Hardware Subsystems

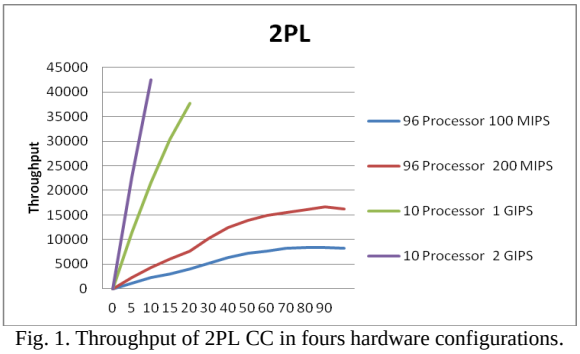

Fig. 1. Throughput of 2PL CC in fours hardware configurations.

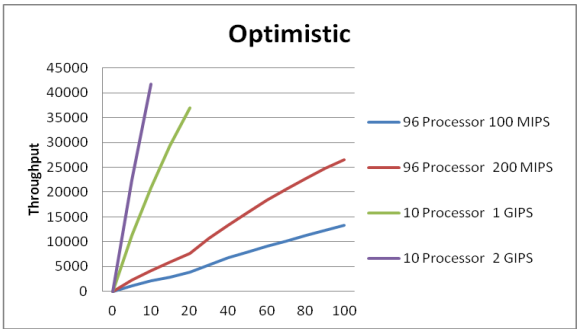

Fig. 2. Throughput of optimistic CC in fours hardware configurations.

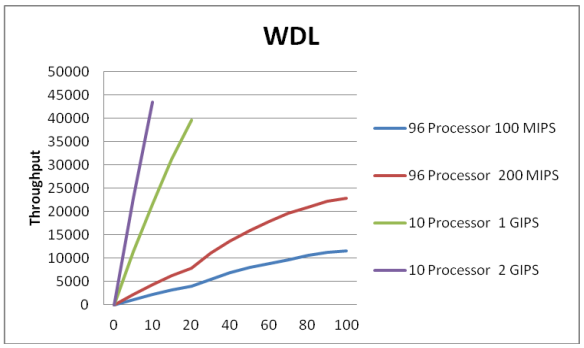

Fig. 3. Throughput of WDL CC in fours hardware configurations.

The charts given in Fig. 1, Fig. 2 and Fig. 3 show the throughputs of each concurrency management scheme by processor configuration of the four hardware subsystems describe above. As the charts show, increase in processor speeds is far more effective than increasing concurrency in increasing throughput under any concurrency management scheme.

B. Throughput Comparison of CC Mechanisms

The charts given in Fig. 4, Fig. 5, Fig. 6 and Fig. 7 provide the comparison of throughputs achieved by the three CC mechanisms for each of the four hardware configurations respectively. As shown in the data of these charts and as would be expected, at low concurrencies, the choice of concurrency management scheme does not have any significant impact on throughput. Thus, in our simulations, differences in performance only appear after a concurrency of 15. At concurrencies between 15 and 50 WDL performs best followed by optimistic. However at higher concurrencies optimistic performs best.

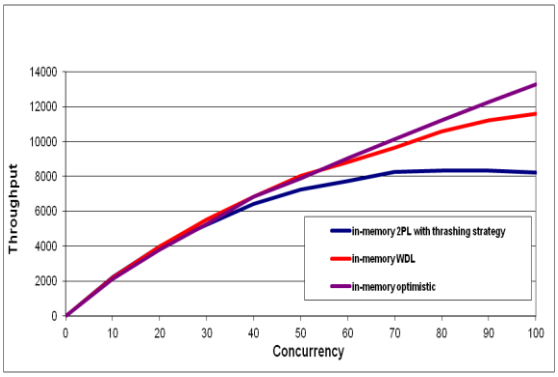

Fig. 4. Throughput comparison of CC mechanisms for 96 processor 100 MIPS.

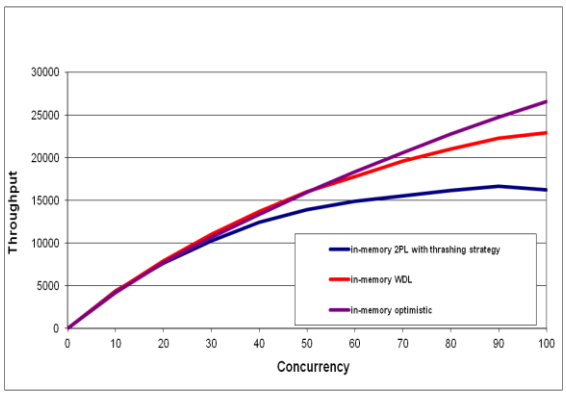

Fig. 5. Throughput comparison of CC mechanisms for 96 processor 200 MIPS.

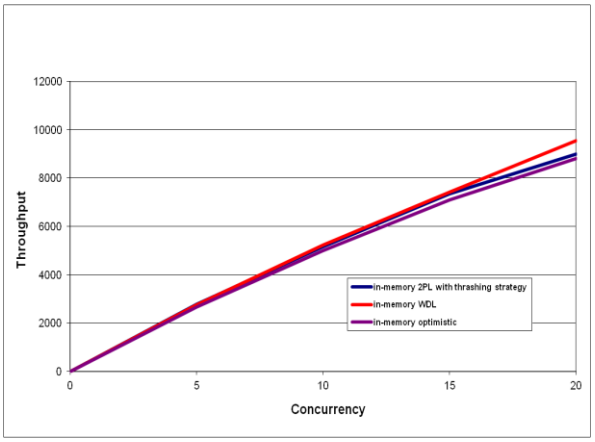

Fig. 6. Throughput comparison of CC mechanisms for 20 processor 1 GIPS.

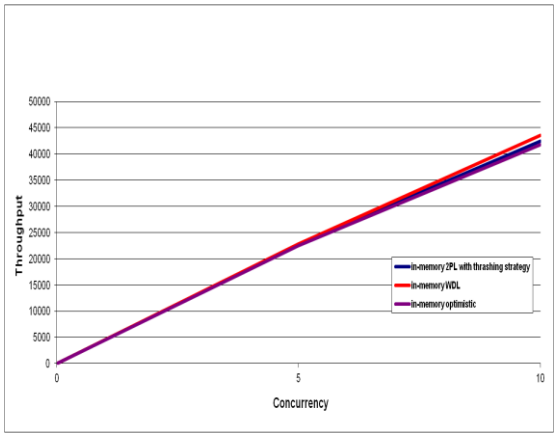

Fig. 7. Throughput comparison of CC mechanisms for 10 processor 2 GIPS.

V. Conclusion and Future Work

In-memory OLTP databases working with a local copy of distributed data schemas provide tremendous advantages over standard shared memory. Apart from secure transactions there is enhanced throughput for high concurrencies. These benefits outweigh their disadvantages of maintenance overhead and timeliness. In addition, over the years, there has been a need to access and process heterogenous and distributed data in different formats warranting parallel or multidimensional high-speed OLTP databases [4], [24], [25]. Optimising concurrency control mechanisms in such environments has been of research interest for more than a decade [26], [27]. In this paper, we investigated three commonly used concurrency methods, 2PL, WDL and Optimistic CC mechanisms and examined their performance relative to each other under diverse hardware configurations for in-memory OLTP databases. The salient points raised by the results presented in this study are:

1) In all our tests, WDL and optimistic concurrency control outperformed 2PL by a wide margin on equivalent hardware at concurrencies of 15 and over. This is consistent with the well-known observation that in systems with abundant hardware capacity, 2PL relatively does not perform well.
2) The rates of throughput achievable by in-memory systems under 2PL, WDL and optimistic concurrency control are massive with throughputs of over 40000 transactions per second being achieved in the fastest of our in-memory systems.
3) While average throughput under 2PL concurrency control in the in-memory systems with fast processors was quite high, this performance was subject to quite large variations even at low concurrencies. Indeed, in some runs, throughput in even the fastest systems was not very much greater than that achieved by the equivalent disk-based systems studied in previous work.
4) Throughputs under all concurrency control mechanisms in the in-memory systems with speeds over 100 MIPS per CPU were well above the best results achieved by any concurrency control method in the disk-based systems tested in previous work.

As the main memory prices are declining with even higher processing speed offered, the results of our paper indicate that for many organizations, the best way to improving the performance of their transaction processing systems lies in switching to in-memory systems rather than improving the performance of their disk-based systems. However, as more OLTP databases get stored almost entirely in memory, the performance of lock managers could become a bottleneck. Though much of existing work has predominantly investigated on single core machine [8], [12], [26], [27], studies on multiple cores competing for access to lock managers operating on very larger databases in multi-tenant cloud environment could be challenging to optimise the OLTP workloads [1], [4], [25], [28], [29]. Hence, this research work comparing three commonly used CC mechanisms with various multi core hardware configurations paves way for further studies in proposing novel CC mechanisms of the future.

In our future work, we would work on investigating time-varying load on the various in-memory resources and perform comparative performance studies in different scenarios such as query-level, transaction-level and application-level.

References

[1] B. Mozafari, C. Curino, and S. Madden, "Resource and performance prediction for building a next generation database cloud," *CIDR*, 2013.
[2] B. Mozafari *et al*., "Performance and Resource Modeling in Highly-Concurrent OLTP Workloads," *Technical report, MIT*, 2013.
[3] M. Ahmad and I. T. Bowman, "Predicting system performance for multi-tenant database workloads," *DBTest*, 2011.
[4] A. Pavl, E. P. C. Jones, and S. Zdonik "Modeling for Optimizing transaction execution in parallel OLTP systems," in *Proc. the VLDB Endowment*, August 27th 31st 2012, Istanbul, Turkey, vol. 5, no. 2, pp. 85-96.
[5] S. Harizopoulos, D. J. Abadi, S. R. Madden, and M. Stonebraker. "OLTP through the looking glass, and what we found there," *SIGMOD*, 2008.
[6] A. Thomson, T. Diamond, P. Shao, K. Ren, S.-C. Weng, and D. J. Abadi, "Calvin: Fast distributed transactions for partitioned database systems," *SIGMOD*, 2012.
[7] P. Larson, S. Blanas, C. Diaconu, C. Freedman, J. Patel, and M. Zwilling, "High-performance concurrency control mechanisms for main-memory database," *PVLDB*, vol. 5, no. 4, pp. 298–309, 2011.
[8] S. Kaspi, "The use of contention-based scheduling for improving the throughput of locking systems," in *Proc. ADBIS 2001*, Vilnius, Lithuania, September 2001.
[9] R. Johnson, I. Pandis, and A. Ailamaki. "Improving OLTP scalability using speculative lock inheritance," *PVLDB*, vol. 2, no. 1, pp. 479–489, 2009.
[10] C. Curino *et al*., "Workload-aware database monitoring and consolidation," *SIGMOD*, 2011.
[11] J. Gray, "Notes on database operating systems. Operating System," *An Advanced Course*, 1979, Berlin: Springer-Verlag.
[12] R. Agrawal, M. J. Carey, and M. Livny, "Concurrency control performance modeling: Alternatives and implications," *ACM Transactions on Database Systems*, vol. 12, no. 14, no. 609–654, 1987.
[13] P. A. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Database Systems*, 1987, Addison-Wesley.
[14] P. A. Bernstein and N. Goodman, "Concurrency control in distributed database systems," *ACM Computing Survey*, vol. 13, no. 2, pp. 185–221, 1981.
[15] D. Agrawal and S. Sengupta. "Modular synchronization in distributed, multiversion databases: Version control and concurrency control," *IEEE TKDE*, vol. 5, 1993.
[16] S. Kaspi, "Optimizing Transaction throughput in databases via an intelligent scheduler," in *Proc. the 1997 IEEE International Conference on Intelligent Processing Systems*, Beijing, October, 1997, pp. 1337–1341.
[17] V. Gottemukkala and T. Lehman, "Locking and latching in a memory-resident database system," *VLDB*, 1992.
[18] I. Pandis, R. Johnson, N. Hardavellas, and A. Ailamaki, "Data-oriented transaction execution," in *Proc. PVLDB*, vol. 3, no. 1, 2010, pp. 928–939.

[19] P. Franaszek and J. T. Robinson, "Limitations of concurrency in transaction processing," *ACM TODS*, vol. 10, no. 1, pp. 1-28, 1985.
[20] A. Thomasian and K. Ryu, "Performance analysis of two-phase locking," *IEEE Transactions on Software Engineering*, vol. 17, no. 5, pp. 386-402, 1991.
[21] P. Franaszek, J. T. Robinson, and A. Thomasian, "Concurrency control for high contention environments," *ACM TODS*, vol. 17, no. 2, pp. 304 - 345, 1992.
[22] A. Thomasian, "Checkpointing for optimistic concurrency control methods," *IEEE Transactions on Software Engineering*, vol. 17, no. 5, pp. 386-402, 1995.
[23] A. Thomasian, "A performance Comparison of locking methods with limited wait depth," *IEEE Transactions on Knowledge and Data Engineering*, vol. 9, no. 3, pp. 421-434, 1997.
[24] C. H. C. Leung, "Parallel paradigms for query evaluation and processing," in *Proc. the Australasian Workshop on Parallel and Real-Time Systems*, PART'94, Melbourne, 1994, pp 1 -10.
[25] S. Venkatraman, "Transforming grid to cloud services for multimodal biometrics," Special Issue on: Urban Computing and Smart Grids', *International Journal of Computational Science and Engineering*, vol. 8, no. 4, pp. 22-34, 2013.
[26] H. T. Kung and J. T. Robinson, "On optimistic methods for concurrency control," *ACM Transactions on Database Systems*, vol. 2, no. 4, pp. 213-226, 1981.
[27] C. H. C. Leung and S. Kaspi, "A flexible paradigm for semantic integration in cooperative heterogeneous databases," presented at FGCS '94, ICOT, Tokyo, December 1994.
[28] V. Ramanathan, S. Venkatraman, and S. R. Asaithambi, "A practical cloud services implementation framework for e-businesses," in K. Tarnay, L. Xu, and S. Imre, Ed., *Research and Development in E-Business through Service-Oriented Solutions*, 2013, IGI Global Publishers, USA.
[29] B. Mozafari, C. Curino, A. Jindal, and S. Madden, "Performance and resource modeling in highly-concurrent OLTP workloads," presented at SIGMOD'13, June 22–27, 2013.

Samuel Kaspi earned his PhD (computer science) from Victoria University, a masters of computer science from Monash University and a Bachelor of Economics and Politics from Monash University. He is a member of both the Australian Computer Society (ACS) and Association for Computing Machinery (ACM).

Sam is currently the Information Technology Discipline Leader and senior lecturer of IT. at the Department of Higher Education - Business in Northern Melbourne Institute of TAFE (NMIT), Australia . Prior to joining NMIT, Dr Kaspi taught at Victoria University, consulted privately and was the CIO of OzMiz Pty Ltd.

Sam has been active in both teaching and private enterprise in the areas of software specification, design and development. As chief information officer (CIO) of a small private company he managed the development and submission of five granted and three pending patents. He also managed the submission of a successful Federal Government Comet grant under the Commercialising Emerging Technologies category. He has also had a number of peer reviewed publications including the Institute of Electrical and Electronics Engineers (IEEE).

Sitalakshmi Venkatraman obtained doctoral degree in computer science, from National Institute of Industrial Engineering, India in 1993 and MEd from University of Sheffield, UK in 2001. Prior to this, she had completed MSc in Mathematics in 1985 and MTech in Computer Science in 1987, both from Indian Institute of Technology, Madras, India. This author is Member (M) of IAENG and Senior Member (SM) of IASCIT.

In the past 25 years, Sita's work experience involves both industry and academics - developing turnkey projects for IT industry and teaching a variety of IT courses for tertiary institutions, in India, Singapore, New Zealand, and more recently in Australia since 2007. She currently works as Lecturer (Information Technology) at the Department of Higher Education - Business in Northern Melbourne Institute of TAFE (NMIT), Australia. She also serves as Member of Register of Experts at Austrlais's Tertiary Education Quality and Standards Agency (TEQSA).

Sita has published seven book chapters and more than 80 research papers in internationally well-known refereed journals and conferences that include *Information Sciences*, *Journal of Artificial Intelligence in Engineering*, *International Journal of Business Information Systems*, and *Information Management & Computer Security*. She serves as Program Committee Member of several international conferences and Senior Member of professional societies and editorial board of three international journals.