The Evolution of TPC-Benchmarks: Why TPC-A and TPC-B are Obsolete

Charles Levine
Tandem Computers

Jim Gray,
Digital Equipment Corporation

Steve Kiss
International Business Machines

Walt Kohler
Digital Equipment Corporation

SFSC Technical Report 93.1, September 1993

San Francisco Systems Center
Digital Equipment Corporation
455 Market St,
San Francisco, CA. 94105

The Evolution of TPC Benchmarks:
Why TPC-A and TPC-B are Obsolete

Charles Levine
Tandem Computers
19333 Vallco Pkwy, LOC 252-10, Cupertino, CA 95014; e-mail: levine_charles@tandem.com

Jim Gray
Digital Equipment Corp.
455 Market St., 7th Floor, San Francisco, CA 94105; e-mail: JimGray@SFbay.enet.dec.com

Steve Kiss
IBM
Hwy 52 & 37th St., NW, Bldg. 006-2, Dept. 51B, Rochester, MN 55901; e-mail: iskiss@rchvmw2.vnet.ibm.com

Walt Kohler
Digital Equipment Corp.
151 Taylor St. (TAY 1), Littleton, MA 01460; e-mail: kohler@tpsys.enet.dec.com

Abstract

TPC benchmarks provide the de facto industry standard for measuring transaction processing performance. This paper argues that TPC-A and TPC-B have served their purpose and are effectively obsolete. Several problems with the benchmarks are described including flaws in the pricing model, the impact of terminals in the price-performance metric, use of client-server configurations, and the use of specialized features to improve benchmark performance. TPC-C addresses many of the problems in TPC-A and B. The advantages of TPC-C are shown by comparison to TPC-A. TPC-A has driven substantial improvements in performance and price-performance, but is at the end of its useful life. TPC-C will drive industry progress in the future.

This paper expands upon an earlier version presented at CompCon 93 [Levine].

Tandem Technical Report 93.1
(Part Number 102034)

Digital San Francisco Systems Center Technical Report TR 93.1

Table of Contents

Introduction

Performance of commercial computing systems is typically measured in *transactions per second* (tps), much as scientific workstation performance is measured in SPECmarks. Price-performance is measured in *dollars per transaction per second* ($/tps). These metrics depend on a standard definition of a transaction. The Transaction Processing Performance Council (TPC) defines standard transaction workloads, including implementation and reporting rules. The TPC also defines system price based on the 5-year capital cost of the associated hardware and software.

The Transaction Processing Performance Council (TPC) is the de facto industry standards body for OLTP performance benchmarks. Its membership includes all the major hardware and software vendors. The TPC's first two benchmark standards, TPC Benchmark™ A (TPC-A) and TPC Benchmark™ B (TPC-B), replaced the ill-defined and often abused Debit/Credit and TP1 benchmarks. They specify precise requirements for the implementation, execution, documentation, and pricing of benchmark results. This established a level playing field on which dozens of vendors now routinely demonstrate the performance and price-performance of their systems.

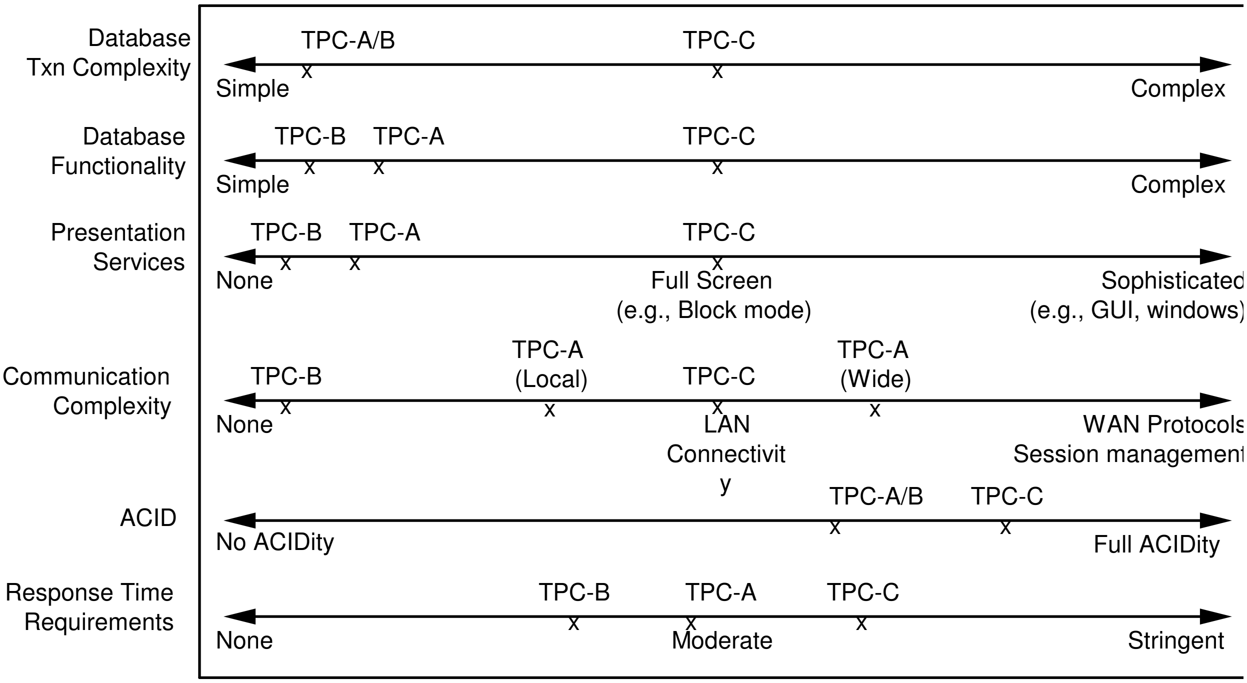
However, advances in technology and the marketplace have made TPC-A and B irrelevant indicators of performance and price-performance for all but a narrow class of OLTP workloads. Since most applications use much more complex transactions than those specified in TPC-A and TPC-B, these benchmarks do not reflect typical application behavior. Fortunately, the recent approval of TPC-C, a complex OLTP benchmark, provides a new venue in which meaningful performance and price-performance comparisons can be made across a wide range of systems.

TPC-C is an evolutionary step forward in TPC benchmarks. Most of the framework and methodology of TPC-C derives from experience with TPC-A; the basic requirements for ACID properties, pricing, full disclosure, and auditing are largely unchanged. TPC-C tests many core database capabilities not covered in TPC-A. On average, TPC-C is an order of magnitude more complex in key areas such as database functionality, transaction complexity, and presentation services. Consequently, TPC-C represents a much broader segment of the OLTP marketplace.

TPC-A and TPC-B Are Too Simple

OLTP systems span a large spectrum of real world applications. These include banking, manufacturing, telecommunications, health care, and finance. Similarly, the workload and performance characteristics of these applications also form a spectrum. The TPC is working to provide benchmarks which cover this spectrum. TPC-A and B generally fall at one end of the spectrum; TPC-C focuses on the center of the spectrum. Figure 1 illustrates aspects of this spectrum.

Figure 1: Characteristics of the OLTP Spectrum



TPC-A and its predecessor, the Debit/Credit benchmark, have been criticized as being too simple to represent real-world OLTP environments. The transaction profile and database schema derive from a simple online banking teller application implemented in 1974. By 1985 it had become a de facto standard [Anon]. The TPC did some "fine-tuning" to make the benchmark more reasonable, and in 1989 the TPC made it a standard de jure.

The unrealistic simplicity of the benchmark's transaction profile was widely appreciated in 1988, and some TPC members were reluctant to standardize it even then. Indeed, the benchmark relies on a single, simple transaction involving only four database tables. Only the simplest form of record access and locking is exercised. Overall, the benchmark stresses a narrow subset of OLTP system functions. Within the OLTP spectrum, TPC-A is a good performance indicator for environments characterized by a static application performing one or two simple transactions with a stringent response time requirement. State lotteries are a good example of this simple workload.

TPC-B defines a database-only benchmark that is more narrowly focused than TPC-A. TPC-B is a subset of TPC-A and thus subject to the same criticisms plus several others. TPC-A and TPC-B share the same transaction profile and database schema. They differ in that TPC-B tests nothing beyond the database, whereas TPC-A includes the terminal network and more strict transaction pacing requirements. Not surprisingly, TPC-B has achieved little industry acceptance and is used far less than TPC-A even though it is much easier for vendors to configure and run. Although this paper focuses primarily on TPC-A, most observations about TPC-A apply equally to TPC-B.

In spite of their drawbacks, TPC-A and B provided a valuable starting point for the TPC. By establishing a solid, well defined benchmark for the industry to follow, the TPC brought to an end the *benchmarketing* games of Debit/Credit and TP1 occurring from 1987 to 1990. For roughly two years, TPC-A provided useful comparisons between systems and vendors.

However within the last year, several factors have combined to greatly diminish the value and relevance of TPC-A results. These are:

- Flaws in the Pricing Model
- Terminals Dominating the Price-Performance Metric
- Use of Ad Hoc Client-Server Configurations
- Use of Specialized Features
- Price-Performance Improvement in TPC-A Outpacing that Seen by Customers
- Emergence of TPC-C

Each of these points is examined in the sections which follow.

Flaws in the Pricing Model

TPC benchmarks use two metrics: performance and price-performance, following the precedent established in Debit/Credit. Unlike the performance metric which is relatively straightforward to specify, the price-performance metric requires detailed pricing rules. TPC benchmarks define a detailed pricing model that includes hardware, software, and maintenance to give an estimate of the five year cost-of-ownership. This approach provides the most meaningful comparison of dissimilar systems because all results are normalized back to price-performance.

The TPC-A pricing model requires standard list pricing to be used. All prices must be generally available to any customer purchasing the same system. Discounts may be used as long as they are generally available and assume no other purchases. Generally available package pricing is allowed.

The TPC pricing model attacked for the first time the issue of quantifying price-performance. However, it suffers from a generic problem: it is easy to reduce the price of something you don't expect anyone to buy. The pricing model does not match discrete purchasing decisions made by customers.

Two significant loopholes have emerged in actual use of the pricing model: package pricing and five year maintenance. Package pricing allows vendors to use legitimate pricing options available to customers to reduce the reported price of the benchmarked system. This is as it should be. Unfortunately, package pricing also opens the door to systems which group together components and options carefully arranged to meet the needs of the TPC-A benchmark, and at the same time be unattractive to real customers. These packages are then steeply discounted to achieve better TPC-A price-performance.

The second loophole has to do with the requirement that five year maintenance be priced. In itself, this is a good requirement. Two opportunities for abuse arise when five years of maintenance is purchased in a single lump sum and prepaid. First, the maintenance can be packaged as a five year service contract. Second, a different price can be offered if the maintenance charges are prepaid. By requiring that a commitment be made to five years of maintenance or that all of the maintenance costs be prepaid, the package becomes very unattractive to most customers. Consequently, the package can be steeply discounted to get better benchmark results without risking any real revenue impact to the vendor.

The problems with maintenance pricing can be fixed by changing the specification. One solution would be to require maintenance be based on monthly or quarterly rates offered by the vendor and disallow any discounts provided for long-term maintenance contracts. The price of five years of maintenance could be computed by extending the monthly or quarterly rate and applying future-value equations to normalize for cost-of-money. This approach has the advantages of pushing vendors to be more competitive on maintenance costs, and allowing customers to more readily compare costs across systems. The disadvantage is that it would arbitrarily preclude a few vendors from using legitimate discounts and maintenance contracts.

Improvements to the pricing rules have been made in TPC-C, although the two loopholes described above still exist. In TPC-C, the pricing model has been improved in the areas of third-party pricing and line-item pricing. The rules for third-party pricing require that any products which are not supplied by the benchmark sponsor must still meet all other pricing requirements and that the pricing must be substantiated by written price quotations. Line-item pricing requires that all pricing be shown to the same level of detail that a customer would see on an itemized bill. Both of these changes were added to TPC-C based on experience from TPC-A.

Customers ultimately have the greatest influence over pricing. Customers using TPC benchmark results to compare systems should also carefully analyze component pricing included in the reports and insist on equal or better pricing for their purchases.

Terminals Dominating the Price-Performance Metric

TPC-A requires that transactions originate from a terminal. Terminals are emulated during benchmark tests, but the pricing model requires that the emulated number of terminals be priced. Since each terminal can submit a transaction at most once every 10 seconds on average, slightly more than 10 terminals per tps must be priced. In addition, all of the terminal connectivity hardware (e.g., terminal servers, controllers) and maintenance must be priced.

Although generally thought of as a keyboard and screen, a "terminal" is any end-user data entry and display device. For example, touch-tone telephones, PCs, ATM machines, cash registers, and interactive cable TVs are all terminals. A terminal may perform only data entry and display, or it may incorporate presentation functions, communication capabilities, encryption, etc.

TPC-A requires terminals that provide only the simplest functions. The terminals must be "commercially available products capable of entering via a keyboard all the alphabetic and numeric characters and capable of displaying simultaneously the data and fields [in the output message]". (TPC-A omits a requirement that the input be echoed to the user as it is entered.) The lowest function dumb terminals satisfy these requirements. More sophisticated terminals provide no benefit but add substantial costs. This has led to vendors using the lowest priced terminals that can be found anywhere in the free world. The current favorite is the Freedom One from Liberty Electronics; $237 each, in quantity. Even recent results from IBM and HP, who sell their own terminals, have priced other manufacturer's terminals to get a lower price.

The original Debit/Credit benchmark on which TPC-A is based had each bank teller think for an average of 100 seconds to enter 100 characters - a fairly typical average think time for such applications. This implied that each transaction per second required 100 teller terminals to generate the load. In 1988, simple terminals cost about $1,000 so this implied a $100,000 base cost per transaction. At that time, the rest of the TP system cost about $50,000/tps. This meant that if unmodified, the Debit/Credit benchmark would imply a 150 K$/tps price-performance metric. Two thirds of the price would be terminals. No one viewed this as a good price-performance metric. Had that design been adopted, today over 90% of the price would be terminals. Instead the TPC postulated super-tellers who are ten times faster than the average. They can think and type a typical transaction in 10 seconds. This reduced the number of terminals to 10 per tps, and at the time made the terminal price less than 20% of the price-performance metric.

Computer prices have fallen substantially in the last few years, but the price of terminals has not fallen nearly as much. Consequently, terminals and terminal connectivity now typically amount to between 40% and 60% of the total price-performance of competitive TPC-A results. So, half of the price-performance in a TPC-A result is determined by the cost of the terminal components. Exactly the problem the TPC wanted to avoid!

The number of terminals in TPC-A is problematic. One advantage of the terminal scaling used in Debit/Credit is that it forced the SUT (System Under Test) to manage a huge number of terminals. With 100+ terminals per tps in Debit/Credit, a transaction monitor was almost certainly required to handle all of the terminal contexts. At 10+ terminals per tps in TPC-A, the number is not sufficient to require the added costs of a TP monitor and wide area network connectivity, and yet still exceeds the number of direct-connected terminals most systems are designed to support. For example, to benchmark a system at 500 TPC-A tps requires at least 5,000 terminals be emulated and priced. Even for small systems, the number of terminals which must be supported to run TPC-A is extreme compared to most customer applications. The extreme simplicity of TPC-A causes this problem.

TPC-C improves on TPC-A by requiring a full screen user interfaces and by increasing the workload generated per terminal. The average TPC-C transaction is between 7 and 10 times heavier than a TPC-A transaction, depending on the system and software used. The average cycle time per transaction is approximately 22 seconds in TPC-C compared with 10 seconds in TPC-A. The net effect of the more complex transaction profile and increased cycle time is to reduce the number of terminals needed by a factor of three to four compared to TPC-A. This provides a better balance between the terminals and other components in the system and presents a more realistic picture of how OLTP systems are really built today.

In addition, TPC-C rewards the use of higher functionality terminals through its user interface requirements. Dumb terminals are not precluded, but they will place substantially greater demands on the host system. Typically, the lower cost of dumb terminals will not make up for the reduced performance of the host. In fact, most TPC-C results released to date have used high functionality, block-mode terminals which range in price from $750 to $1150 each, compared to the $237 dumb terminal commonly used in TPC-A. Unfortunately, because the decrease in the number of terminals roughly equals their increased cost, terminals already comprise about 30% to 50% of TPC-C price-performance.

Use of Ad Hoc Client-Server Configurations

The majority of TPC-A results now being published use what is referred to as a client-server configuration. In the real world, client-server computing typically means that each user has an intelligent device such as a PC or workstation which utilizes high performance networks to off-load work from the host and provide sophisticated presentation services.

In TPC-A benchmarks, "client-server" configurations consist of PCs or workstation front-ends to which are connected the large number of terminals required by TPC-A. (See figure 2.) These front-end systems serve primarily as multiplexors to off-load the terminal contexts and optimize the performance of the more expensive back-end systems. Although this complies with the rules of TPC-A (since the front-end systems are included in the price), the resulting benchmark performance represents neither a classical terminals-connected-to-host configuration nor a modern client-server architecture.

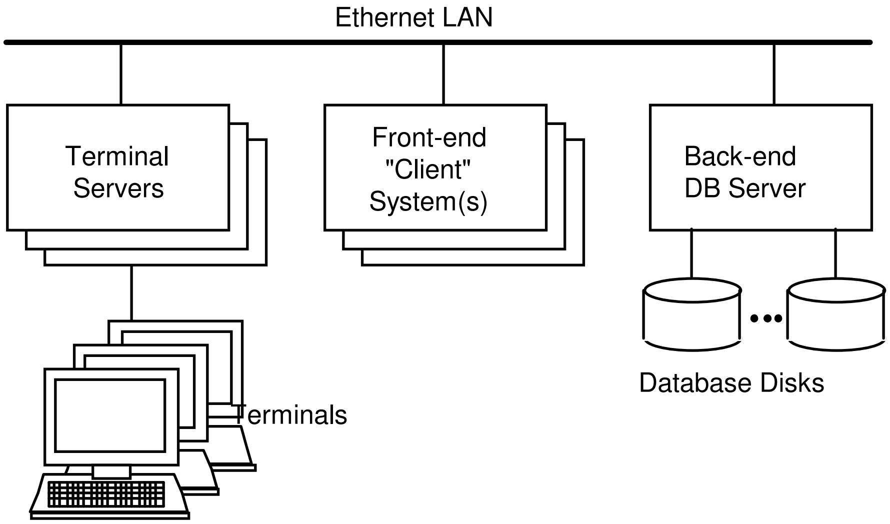


Figure 2: Typical "client-server" configuration used in TPC-A benchmarks

Two objections can be raised against many of the hybrid configurations used in TPC-A benchmarks:

- They are ad hoc arrangements of hardware and software.
- They obscure the performance of the individual components being used.

The ability to off-load processing to more economical front-end systems is clearly important to customers. This allows customers to reduce overall system cost and increase the capacity of back-end systems. This is quite reasonable when it is part of an architected and supported software solution. Many client-server configurations used in TPC-A results, though, are ad hoc arrangements of hardware and software. They are not robust products viable for customer environments, but rather a collection of components assembled for the benchmark. Client-server configurations need to be supported as a unified system, via the development environment, tools, and administrative facilities. Application development and system management become more complex in client-server environments and these must be addressed for the whole system to provide a viable solution.

The second objection is that hybrid configurations obscure the performance of the individual components being used. It is not possible to determine how much the performance of the back-end database system is being improved by work off-loaded to the front-end. This makes such results simultaneously less useful and more misleading. For example, many vendors have published TPC-A results in which a high performance system is front-ended by numerous lower cost machines. In marketing the results, vendors focus on the back-end system, silently overlooking the work off loaded to the low cost front-end machines. The TPC now requires that client-server configurations be labeled "c/s" to distinguish them from single machine results. The TPC needs to address the larger problem by requiring that more performance data be disclosed so that the distribution of performance in client-server configurations can be readily determined.

TPC-C provides more opportunities to benefit from client-server configurations compared to TPC-A because of the expanded terminal presentation services required in the benchmark. In fact, a TPC-C benchmark using PCs or workstations (in place of terminals) as clients in a true client-server environment is a very reasonable implementation except that the price of the PCs is prohibitive compared to terminals. Client-server configurations such as pictured in Figure 2 will most likely become at least as common in TPC-C as they are today in TPC-A.

Use of Specialized Features

A good benchmark does not lend itself to special-case optimizations. Ideally, a benchmark exercises representative features and capabilities so that any improvement made in the performance of the benchmark provides a corresponding improvement for real-world applications. And yet, this is frequently not the case. Standardized benchmarks put tremendous pressure on vendor marketing and development to improve performance and reduce price regardless of whether these changes will benefit actual customers. This has a double negative effect. Since development resources are always limited, effort spent on benchmark features uses resources in ways that do not benefit customers and reduces the resources available for customer oriented work.

As mentioned previously, OLTP systems span a large spectrum of real world applications. TPC-A models a narrow slice of this spectrum. Products that perform well at one end of the spectrum may do poorly in a different application. Although benchmarks like TPC-A show the performance of only a single workload, vendors generally infer that the performance (if good) is representative across a wide range of workloads. In fact, benchmark performance becomes less representative of overall performance in direct proportion to any specialized features used to implement the benchmark.

Perhaps it is a measure of the TPC's success that vendors are now introducing special features into their products which optimize TPC-A performance. A recent example of this is a feature in Oracle version 7.0 called *discrete transactions*. Designed to improve the performance of short, non-distributed transactions within a highly restricted environment, discrete transactions provide an optimized path for TPC-A and B benchmarks.[1] (For example, some of the restrictions placed on discrete transactions are: no database block can be modified more than once per discrete transaction; a discrete transaction cannot see the new value of the data after modifying it; discrete transactions cannot be used in distributed transactions; and discrete transactions cannot be used with some referential integrity constraints. [Oracle]) Similar to the field-call feature in IMS FastPath, discrete transactions improve the performance of a specialized case. This is a classic trade-off of generality versus performance.

Specialized features narrow the applicability of benchmark results and can impugn the results' comparability [Standish]. In the worst case, a benchmark result indicates nothing more than the performance which can be achieved on that specific workload. One recent article [Moad] suggested TPC-A has become "a vendors' proving ground for new techniques in predatory marketing and exotic engineering". Benchmark results which achieve performance improvements using specialized features draw unfair comparisons between systems that customers perceive to be comparable. Performance experts can draw this distinction by carefully examining the benchmark implementation provided in the full disclosure report, but this defeats the purpose of standardized benchmarks. The question which customers need to ask is whether the benchmark provides a valid comparison using features and capabilities that matter to them.

The TPC can address the problem of benchmark specials in two ways. First, benchmark specials can be explicitly prohibited in the specs. Unfortunately, any wording which specifies what constitutes a benchmark special will be subjective in nature and could be difficult to enforce. There may be a fine line between a benchmark special and an innovative new technology. Some TPC members have argued that outlawing benchmark specials stifles new technology. In practice, though, useful technology is distinguished from a benchmark special by the real world customers using it.

The second way the TPC can address benchmark specials is to have better benchmarks. A fundamental problem of TPC-A is that it is too simple. This simplicity lends itself to specialized optimizations. Because TPC-C is a much broader benchmark, it will be less prone to optimizations from specialized features. In the same way, changes made to improve TPC-C performance will be much more likely to also benefit a wide range of customer workloads.

Price-Performance Improvement in TPC-A Outpacing that Seen by Customers

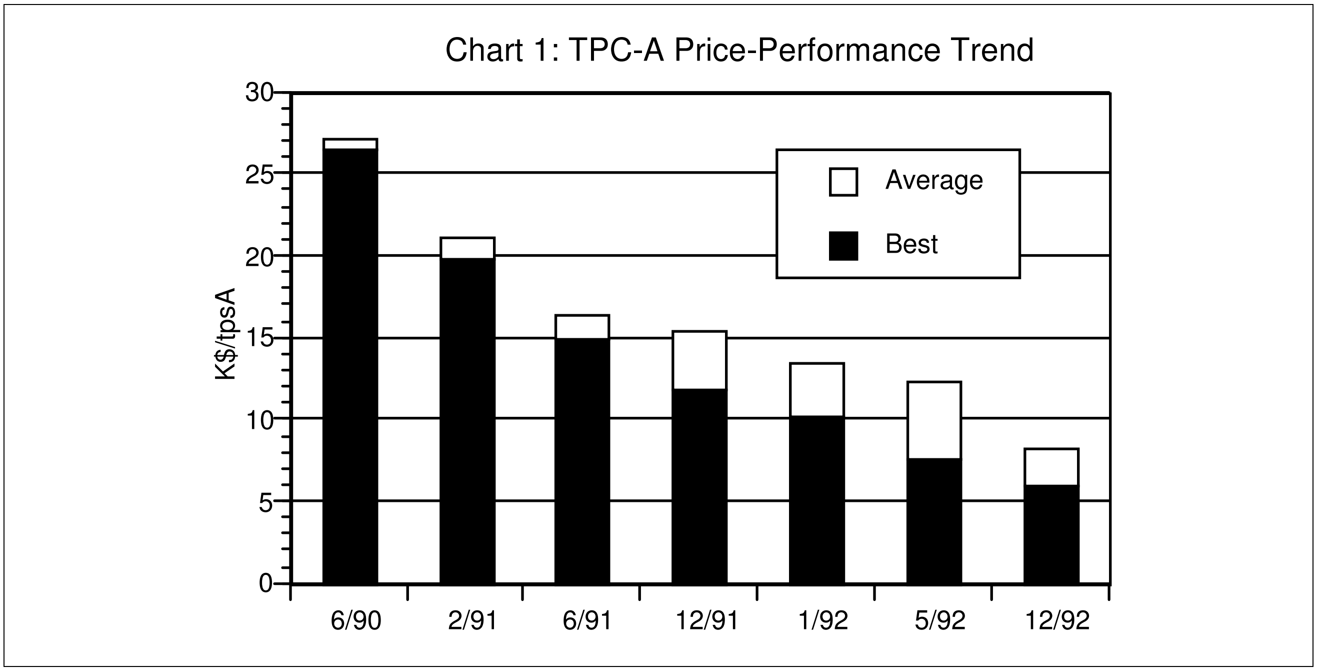
Between 1989 and 1993, TPC-A price-performance went from 33 K$/tps to 6 K$/tps, and throughput passed 1000 tps. This dramatic improvement in throughput and price-performance reflects three forces: (1) the commoditization of hardware - prices typically dropped by a factor of four, (2) tuning products to perform well on the benchmark, and (3) the introduction of TPC specials in software and package pricing.

When the standard was approved, each database vendor modified its system to run TPC-A well. Many systems improved by a factor of two or four. Some vendors eliminated bottlenecks, and so got huge improvements. Many of these improvements are generic (e.g., write-ahead log, group commit, record locking, etc.) so that conventional applications have benefited from them.

[1] Note: TPC-A and B results using Oracle7 are accepted by the TPC as fully complying with the requirements of these benchmarks. No implication to the contrary is intended.

But today, most systems have fixed their "general" TPC-A problems and there is an alarming trend to implement special features to get the next leg up in the TPC-A race.

Chart 1 shows TPC-A price-performance for the last three years. TPC-A price-performance has improved approximately 35% per year for the average of all published TPC-A results. The best, or leading edge, TPC-A results have improved approximately 43% per year. This rate of improvement significantly exceeds that seen by customers.



The price-performance reported by TPC benchmarks reflects only a fraction of the actual cost of ownership. Not included in TPC benchmarks is the cost of facilities, development staff, operation management, project risk, and system growth (or upgrades). These factors increase cost of ownership by two to three times.

In the real world, many factors contribute to improved price-performance seen by customers. Some of the significant factors are:

- Reduced hardware, software, and maintenance prices
- Increased hardware performance (speed, parallelism, architecture)
- Improved software performance (shorter path length, more parallelism)

These factors are fully reflected in TPC-A results. In addition, however, several factors exaggerate price-performance improvements shown by TPC-A results:

- Improvements designed to help short, high-frequency transactions.
- Changes in benchmark implementation to optimize performance.
- Special features which take advantage of the simplicity of TPC-A.
- Pricing games which provide steep discounts for packages real customers will never buy.

Hardware and software improvements designed to help simple transactions in a well controlled environment will have a major impact on TPC-A results, while hardly being noticeable to most customers. Optimizing the benchmark implementation also provides significant improvements. This is valid in that customers optimize implementation of critical applications, but in this case, improvements in the benchmark implementation appear as system improvements. Furthermore, benchmark implementation improvements may take advantage of the benchmark's simplicity in ways that customers would not be able to with real applications. The simplicity of TPC-A invites 'TPC specials' in system implementation. Specialized features which take advantage of the limited nature of TPC-A improve performance, but may be useless to most customers. Finally, pricing tricks used in TPC benchmarks exaggerate price-performance improvements beyond that seen by customers.

An ideal benchmark would show the same price-performance improvement that a real customer would see. This is of course impossible to achieve, because a benchmark is only a model of the customer environment, and thus cannot include all the elements of the customer operation. TPC-A was designed to model a subset of the OLTP environment, dominated by simple transactions. Increased realism in TPC-C will hopefully significantly reduce, though not totally eliminate, this problem.

Emergence of TPC-C

TPC-C is the TPC's first "made-from-scratch" benchmark. TPC-A and B closely resemble the Debit/Credit and TP1 benchmarks that were in widespread use at the TPC's founding. The TPC merely formalized the benchmarks. In TPC-C, the TPC crafted a new benchmark [Raab]. TPC-C makes major improvements over TPC-A in the following areas:

- complexity of database schema
- complexity of transaction mix
- variability in transaction weight
- location transparency for all database accesses
- realistic user interface
- realistic data access patterns
- use of database functionality

Each of these points is examined below.

The database schema used in TPC-C is richer and more complex than TPC-A's. (See Table 1 below.)

Table 1: Comparison of Database Schemata between TPC-A and TPC-C

	TPC-A	TPC-C
Number of Tables	4	9
Indexes required (in addition to primary key access path)	0	2
Average columns per table	4 (min 3, max 6)	10 (min 3, max 21)
Column types	numeric	numeric, char, varchar, and date-time
Database size	~150 MB/tpsA	~75 MB/tpmC
Relative database size vs. TPC-A (depends on TPC-A to TPC-C performance ratio)	1.0	1.8 - 2.0

TPC-C is a mix of five transaction types: New-Order, Payment, Delivery, Order-Status, and Stock-Level. The characteristics of each transaction are shown in the table below. The TPC-A transaction is included for comparison. As can be seen in Table 2, the five transactions are very different in the amount of work each performs. Variability in the transaction workload is very common in complex OLTP applications and has a major impact on a system's overall performance. The TPC-C workload contains substantial variability in both the mix of transactions and within the transactions themselves. In contrast, TPC-A contains no variability.

Table 2: Comparison of Transaction Workloads between TPC-A and TPC-C

Transaction	Description	Relative Frequency (%)	Number of Tables Accessed	Logical I/O's (average)	Avg. Weight vs. TPC-A txn	% of CPU workload†
TPC-A txn	light OLTP	100%	4	5	1	100%
New-Order	heavy OLTP	43.5	8	46	10	53
Payment	light OLTP	43.5	4	8	2	11
Delivery	mid-weight, "batch" txn	4.4	4	70	15	8
Order-Status	light query	4.4	3	12	1	1
Stock-Level	heavy query	4.4	3	401	50	27

† these are approximations and will vary for each system

TPC-C requires complete location transparency for all database accesses. The DBMS must hide all aspects of physical data distribution from the application. All data must be uniformly accessible regardless of its physical location or storage characteristics. No exceptions are made for distributed systems.

TPC-C specifies a detailed user interface. Each transaction has a full screen layout for input and output data based on a 24 line by 80 character display. Although this does not represent the leading edge in client-server technology, it is representative of a large fraction of the existing installed base for OLTP applications. Furthermore, a full screen user interface is a major step beyond TPC-A, which does not specify the user interface at all.

TPC-C simulates both uniform and skewed data access (commonly known as the 80-20 or 90-10 rule, i.e., 80% of the accesses are to 20% of the data). Both uniform and skewed access patterns occur in real world environments. Since the data access pattern affects cache behavior and database contention, exercising both kinds of access has a significant influence in the benchmark.

TPC-C exercises a broad range of database functionality. All of the core SQL operations are used: update, insert, and delete. In addition, most implementations use cursors and aggregate functions. The stock-level query performs a multi-table join. Transaction rollback (abort) occurs for 1% of the new-order transactions. None of these functions are exercised in TPC-A.

Ultimately, TPC-C is the linchpin to the TPC's continued progress. TPC-C advances the TPC's mission to provide objective performance data relevant to a broad segment of the industry. It addresses many of the shortcomings of TPC-A and provides the foundation for future TPC benchmark development.

Conclusion

TPC-A has served its purpose. It established the playing field on which vendors demonstrate OLTP performance. It provided the vehicle by which the TPC became a viable entity serving the industry. And it has driven the industry to improve products and be much more competitive in price-performance. But it has outlived its usefulness.

Benchmarks have a shelf life. TPC-A has driven development forward until recently, but it is no longer exerting a positive influence on product development. Because vendors adapt their products to perform well on particular benchmarks, the benchmarks themselves must evolve if they are to continue to drive performance and technological innovation. TPC-A could be revised to address some of the problems described above. The problem is that small changes will not help, and large changes will sacrifice the existing set of benchmark results while still not fundamentally addressing TPC-A's technical shortcomings.

The TPC-A workload represents a point on a spectrum. Most applications use much more complex transactions, so TPC-A is unrepresentative of most real world environments. Because of pricing tricks and specialized features, TPC-A results have become less representative of overall performance and less comparable. This derives in large measure from TPC-A being too simple. TPC-C substantially improves on TPC-A by exercising more aspects of a system and addressing many of the problems which have occurred with TPC-A.

TPC-C represents a broad segment of the OLTP spectrum. For benchmarks to be useful, customers must understand how particular benchmarks are relevant to their environment. TPC-A is at the end of its useful life to customers and the industry because it lacks any broad customer relevance. In contrast, TPC-C measures a robust workload using a complex database, multiple transaction types, broad database functionality, and a realistic user workload. Most customers who examine the TPC-C workload will find key elements familiar from their own environment.

Advances in technology and customer requirements argue that the time has come for the industry to focus on a better benchmark such as TPC-C. Because a benchmark must limit diversity to be useful, there is no such thing as a perfect benchmark. Nevertheless, TPC-C greatly advances the state-of-the-art in OLTP benchmarking. Many of TPC-A's problems are solved by TPC-C. Customers can look forward to TPC-C results setting the standard in OLTP performance over the next few years.

Appendix A: What is the TPC?

The TPC has evolved tremendously in the four years since its founding, growing from the original eight vendors to 42 member companies today. The membership includes all the major computer hardware and software companies, two end-user organizations, and two auditing corporations. The TPC is actively defining new benchmarks to expand the scope and relevance of TPC results to the industry. Furthermore, the TPC actively supports its standards through established internal procedures to settle challenges to benchmark results and resolve questions of interpretation in the standards. ([Serlin] provides a history of the TPC.)

Appendix B: Description of TPC-A and TPC-B

The TPC Benchmark A Standard [TPC-A] provides the following description of the benchmark:

TPC Benchmark A is stated in terms of a hypothetical bank. The bank has one or more branches. Each branch has multiple tellers. The bank has many customers, each with an account. The database represents the cash position of each entity (branch, teller, and account) and a history of recent transactions run by the bank. The transaction represents the work done when a customer makes a deposit or a withdrawal against his account. The transaction is performed by a teller at some branch.

TPC Benchmark A exercises the system components necessary to perform tasks associated with that class of on-line transaction processing (OLTP) environments emphasizing update-intensive database services. Such environments are characterized by:

- *Multiple on-line terminal sessions*
- *Significant disk input/output*
- *Moderate system and application execution time*
- *Transaction integrity*

The metrics used in TPC Benchmark A are throughput as measured in transactions per second (tps), subject to a response time constraint; and the associated price-per-tps. This benchmark uses a single, simple, update-intensive transaction to load the system under test (SUT). Thus the workload is intended to reflect an OLTP application, but does not reflect the entire range of OLTP requirements typically characterized by multiple transaction types of varying complexities. The single transaction type provides a simple, repeatable unit of work, and is designed to exercise the key components of an OLTP system. The transaction profile is shown below:

```
Read 100 bytes including Aid, Tid, Bid, Delta from terminal.
BEGIN TRANSACTION
   Update Account where Account_ID = Aid:
         Read Account_Balance from Account
         Set Account_Balance = Account_Balance + Delta
         Write Account_Balance to Account
   Write to History:
         Aid, Tid, Bid, Delta, Time_stamp
   Update Teller where Teller_ID = Tid:
         Set Teller_Balance = Teller_Balance + Delta
         Write Teller_Balance to Teller
   Update Branch where Branch_ID = Bid:
         Set Branch_Balance = Branch_Balance + Delta
         Write Branch_Balance to Branch
COMMIT TRANSACTION
Write 200 bytes including Aid, Tid, Bid, Delta, Account_Balance to terminal.
```

Aid (Account_ID), Tid (Teller_ID), and Bid (Branch_ID) are keys to the relevant records/rows.

In the case of TPC-B, the description of TPC-A applies except that TPC-B is designed only to exercise the basic components of a database system. TPC-B is specifically not an OLTP benchmark in that no terminals or networking components are included. All transaction inputs and outputs are local to the SUT. Unlike TPC-A, TPC-B does not require a separate driver system.

Appendix C: Description of TPC-C

The TPC Benchmark C Standard [TPC-C] provides the following description of the benchmark:

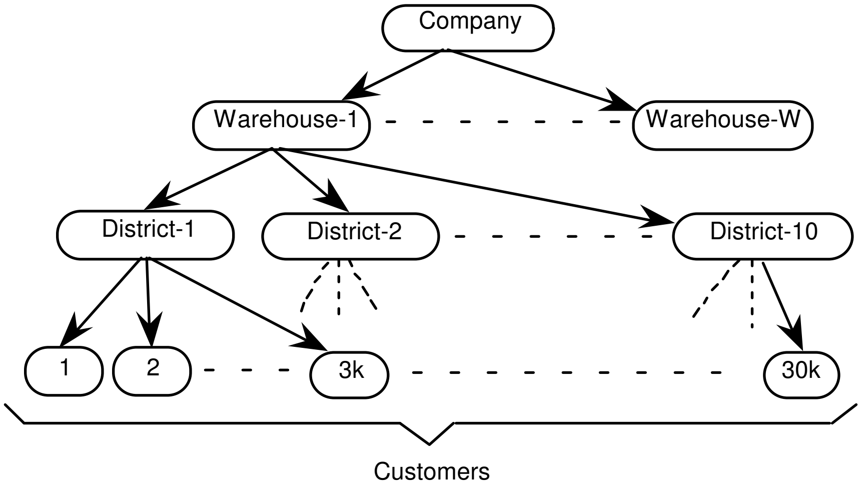
TPC Benchmark™ C (TPC-C) is an OLTP workload. It is a mixture of read-only and update intensive transactions that simulate the activities found in complex OLTP application environments. It does so by exercising a breadth of system components associated with such environments, which are characterized by:

- *The simultaneous execution of multiple transaction types that span a breadth of complexity*
- *On-line and deferred transaction execution modes*
- *Multiple on-line terminal sessions*
- *Moderate system and application execution time*
- *Significant disk input/output*
- *Transaction integrity (ACID properties)*
- *Non-uniform distribution of data access through primary and secondary keys*
- *Databases consisting of many tables with a wide variety of sizes, attributes, and relationships*
- *Contention on data access and update*

The performance metric reported by TPC-C is a "business throughput" measuring the number of orders processed per minute. Multiple transactions are used to simulate the business activity of processing an order, and each transaction is subject to a response time constraint. The performance metric for this benchmark is expressed in transactions-per-minute-C (tpmC).

TPC Benchmark™ C is comprised of a set of basic operations designed to exercise system functionalities in a manner representative of complex OLTP application environments. The workload is centered around the activity of processing orders and provides a logical database design, which can be distributed without structural changes to transactions. These *basic operations have been given a life-like context, portraying the activity of a wholesale supplier, to help users relate intuitively to the components of the benchmark.*

The Company portrayed by the benchmark is a wholesale supplier with a number of geographically distributed sales districts and associated warehouses. As the Company's business expands, new warehouses and associated sales districts are created. Each regional warehouse covers 10 districts. Each district serves 3,000 customers. All warehouses maintain stocks for the 100,000 items sold by the Company. The following diagram illustrates the warehouse, district, and customer hierarchy of TPC-C's business environment.



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). One percent of all order lines are for items not in-stock at the regional warehouse and must be supplied by another warehouse. The Company's system is also used to enter payments from customers, process orders for delivery, and examine stock levels to identify potential supply shortages.

References

[Anon] Anon et al., A measure of transaction processing power, *Datamation*, p. 112, April 1, 1985: Vol. 31, No. 7.

[Levine] Levine, Charles, Why TPC-A and TPC-B Are Obsolete, *Digest of Papers, CompCon 93*, IEEE Computer Society Press, Los Alamitos, CA, February 22, 1993, p. 215.

[Moad] Moad, Jeff, Benchmarks Get Real, *Datamation*, p. 61, December 15, 1992: Vol. 38, No. 25.

[Oracle] Oracle, Discrete Transactions, *Oracle7 Server Application Developer's Guide*, p. A-5, December, 1992, Part Number 6695-70-1292.

[Raab] Raab, Francois, Overview of the TPC Benchmark C, *TPC Quarterly Report*, p. 1, April 15, 1992, Vol. 1, Issue 1.

[Serlin] Serlin, Omri, The History of DebitCredit and the TPC, *The Benchmarking Handbook for Database and Transaction Processing Systems, Second Edition*, p. 21, Edited by Jim Gray, Morgan Kaufman Publishers, Inc., San Mateo, CA, 1993.

[Standish] The Standish Group, Foul Flag Flies for Oracle 7, TPC, *Open OLTP Report/The Standish Compact*, p.2, Vol. 4, No. 3, 1993. Published by The Standish Group International, Inc., 295 White's Path, S. Yarmouth, MA 02664.

[TPC-A] *TPC Benchmark A Specification*, The Transaction Processing Performance Council, 1992. Administered by Shanley Public Relations, 777 N. First St., Suite 600, San Jose, CA 95112.
(Also in *The Benchmarking Handbook...*; see [Serlin] above.)

[TPC-C] *TPC Benchmark C Specification*, The Transaction Processing Performance Council, 1992. Administered by Shanley Public Relations, 777 N. First St., Suite 600, San Jose, CA 95112.
(Also in *The Benchmarking Handbook...*; see [Serlin] above.)