TPC-E vs. TPC-C: Characterizing the New TPC-E Benchmark via an I/O Comparison Study

Shimin Chen[1], Anastasia Ailamaki[2], Manos Athanassoulis[2], Phillip B. Gibbons[1]
Ryan Johnson[3], Ippokratis Pandis[3], Radu Stoica[2]

[1]*Intel Labs Pittsburgh* [2]*École Polytechnique Fédérale de Lausanne* [3]*Carnegie Mellon University*

ABSTRACT

TPC-E is a new OLTP benchmark recently approved by the Transaction Processing Performance Council (TPC). In this paper, we compare TPC-E with the familiar TPC-C benchmark in order to understand the behavior of the new TPC-E benchmark. In particular, we compare the I/O access patterns of the two benchmarks by analyzing two OLTP disk traces. We find that (i) TPC-E is more read intensive with a 9.7:1 I/O read to write ratio, while TPC-C sees a 1.9:1 read-to-write ratio; and (ii) although TPC-E uses pseudo-realistic data, TPC-E's I/O access pattern is as random as TPC-C. The latter suggests that like TPC-C, TPC-E can benefit from SSDs, which have superior random I/O support. To verify this, we replay both disk traces on an Intel X25-E SSD and see dramatic improvements for both TPC-C and TPC-E.

1. INTRODUCTION

On-Line Transaction Processing (OLTP) is used extensively to support the daily operations of a wide range of businesses, from banking and grocery stores, to online ECommerce web sites and financial markets. Because of its importance, OLTP has been a major optimization target for computer manufacturers, database software vendors, system software vendors, and the corresponding research communities.

There are two standard TPC benchmarks for OLTP. Since 1992, TPC-C [6] has been the primary benchmark for evaluating OLTP performance. Both industry and the research community have gained deep understandings of TPC-C from many years of studies. In February 2007, the new TPC-E benchmark [7] became a TPC standard. It is designed to be a more realistic OLTP benchmark than TPC-C, e.g., incorporating realistic data skews and referential integrity constraints. However, TPC-E is much more sophisticated than TPC-C. Partly because of this, there is a lack of in-depth understandings of TPC-E, potentially slowing down the adoption of the benchmark.

In this paper, we study the TPC-E benchmark. We take a comparison approach: we compare the TPC-E benchmark with the familiar TPC-C benchmark to identify their similarities and differences. This approach is beneficial in that (i) it takes advantage of our existing knowledge of TPC-C to facilitate the understanding of TPC-E; and (ii) the comparison results show where existing techniques can be applied effectively to TPC-E and where new optimizations may be needed.

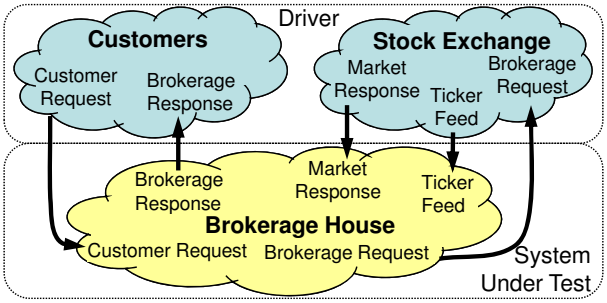


Figure 1: TPC-E models a financial brokerage house. (Figure is adapted from [8].)

We focus on the I/O behaviors of the benchmarks because I/Os are often the major performance aspect in OLTP systems. We use a TPC-E disk trace and a TPC-C disk trace obtained on a commercial DBMS running on medium-sized computer systems with hundreds of disks [4]. We compare various I/O characteristics of the two traces, including request types, sizes, spatial and temporal locality. We find that TPC-E is more read intensive than TPC-C, and its I/O access pattern is as random as TPC-C, though TPC-E is designed to have more realistic data skews. The latter suggests that like TPC-C, flash-based Solid State Drives (SSDs) may significantly improve the performance of TPC-E. We verify this point by replaying the OLTP traces on an SSD.

The rest of the paper is organized as follows. Section 2 provides a high level overview of the TPC-E benchmark, and compares the major features of TPC-E with those of TPC-C. Section 3 analyzes the two OLTP disk traces to compare the I/O access patterns of TPC-E and TPC-C. Section 4 reports the experimental results of replaying disk traces using a state-of-the-art SSD. Finally, Section 5 concludes the paper.

2. TPC-E OVERVIEW

The TPC-E benchmark models a financial brokerage house, as shown in Figure 1. There are three compo-

nents: customers, brokerage house, and stock exchange. TPC-E's focus is the database system supporting the brokerage house, while customers and stock exchange are simulated to drive transactions at the brokerage house.

In the TPC-E database, the brokerage house maintains information for customers (e.g., accounts, holdings, watch lists), brokers (e.g., trades, trade history), and the financial markets (e.g., companies, securities, related news items, last trades). There are two main types of transactions in TPC-E: customer-initiated transactions and market-triggered transactions, as shown in Figure 1. In customer-initiated transactions, customers send requests to the brokerage house, which then queries or updates its database, submits brokerage requests to the stock exchange, and/or returns response to the customers. TPC-E supports both immediate and limit trading orders. For the former, the brokerage house sends brokerage requests immediately. For the latter, it records the orders and their limit prices. In market-triggered transactions, the stock exchange markets send trade results or real-time market ticker feeds to the brokerage house. The brokerage house updates its database, checks the recorded limit trading orders, and submits such orders if the limit prices are met.

Table 1 compares TPC-E and TPC-C. First, their underlying application models are different: TPC-E models a financial brokerage house, while TPC-C models a more traditional wholesale supplier.

Second, there are 33 tables in TPC-E, over three times as many as in TPC-C. Of the 33 TPC-E tables, there are 9 tables recording customer account information, 9 tables recording broker and trade information, 11 market-related tables, and 4 dimension tables for addresses and fixed information such as zip codes. 19 of the 33 tables scale as the number of customers, 5 tables grow during TPC-E runs, and the remaining 9 tables are static.

Third, TPC-E has twice as many columns as TPC-C. However, a TPC-E table has on average 5.7 columns, about half as many as TPC-C. Therefore, although TPC-E includes many more tables for more sophisticated entity relationships, the structures of individual TPC-E tables are relatively simpler.

Fourth, TPC-E has twice as many transaction types as TPC-C. TPC-E has 6 read-only and 4 read-write transaction types, compared to 2 read-only and 3 read-write transaction types in TPC-C. 76.9% of the generated transactions in TPC-E are read-only, while only 8% of TPC-C transactions are read-only. This suggests that TPC-E is more read intensive than TPC-C.

Fifth, the TPC-E database is populated with pseudo-real data that are based on the year 2000 U.S. and Canada census data and actual listings on the NYSE and NASDAQ stock exchanges. In this way, TPC-E reflects natural data skews in the real world. This addresses the complaint of TPC-C using random data that do not reflect real-world data distributions.

Table 1: Comparing TPC-E and TPC-C features.

	TPC-E	TPC-C
Business model	Brokerage house	Wholesale supplier
Tables	33	9
Columns	188	92
Columns/Table	2–24, avg 5.7	3–21, avg 10.2
Transaction mix	4 RW (23.1%) 6 RO (76.9%)	3 RW (92%) 2 RO (8%)
Data generation	Pseudo-real, based on census data	Random
Check constraints	22	0
Referential integrity	Yes	No

Note: The table is based on [6, 7, 8].

Finally, TPC-E incorporates several features that are found in real-world OLTP applications but missing in TPC-C, such as check constraints and referential integrity.

In summary, TPC-E is a more sophisticated, more realistic OLTP benchmark than TPC-C. However, its test setup is more complicated, requiring the development of customer and market drivers. To reduce such efforts, TPC-E distributes code for the core driver logic and describes example SQL statements for implementing most TPC-E transactions. Nevertheless, the sophistication and the lack of in-depth understandings of the TPC-E benchmark may slow the adoption of the benchmark in industry and the research community.

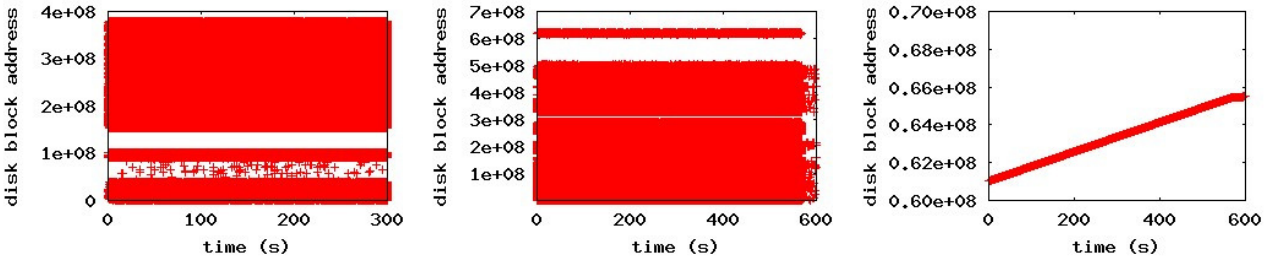
3. COMPARING TPC-E AND TPC-C USING OLTP DISK TRACES

In this section, we compare the I/O behaviors of the TPC-E benchmark and the familiar TPC-C benchmark using two OLTP disk traces obtained on a commercial DBMS [4][1]. We first describe the two OLTP traces, then compare the I/O characteristics of the two traces.

Trace Description. The TPC-C trace was obtained on a 4 dual-core 3.4 GHz Xeons system with 64GB memory and 392 15Krpm SCSI drives organized into 14 RAID-0 disk arrays of 28 disks each. The TPC-E trace was obtained on a 4 quad-core 1.8 GHz Opterons system with 128 GB memory and 336 15Krpm SCSI drives organized into 12 RAID-0 disk arrays of 28 disks each. The TPC-C run was configured with 14,000 warehouses and 300 users. The TPC-C trace is about 5 minutes long. The TPC-E run was configured with 200,000 customers. The TPC-E trace is about 10 minutes long.

The traces contain mainly I/O events and a few other event types (such as thread, process, and context switch events). Information about I/O events include device IDs, offsets, sizes, request types, start and elapsed times.

[1]Kavalanekar et al. [4] described high-level characteristics of 14 storage traces, including the OLTP traces. Here, we take a database centric view to analyze the OLTP traces in depth.



(a) A data device in TPC-C trace (b) A data device in TPC-E trace (c) The log device in TPC-E trace

Figure 2: Spatio-temporal graphs of the TPC-C and the TPC-E disk traces.

We estimate the database size by counting the number of 1MB units that see at least one I/O request: The TPC-C database is 1.5 TB large, and the TPC-E database is 1.7 TB large. The sizes are roughly comparable. Moreover, the I/O rates of the two traces are also roughly comparable, as will be shown in Figure 3.

Spatio-Temporal Graphs. We begin our study by looking at the two dimensional spatio-temporal graphs of the traces. The trace events contain device IDs, corresponding to disk arrays. For each device, we generate a spatio-temporal graph as shown in Figure 2. Every (x, y) point in the figure represents an I/O access to (512-byte) block address y issued to the logical device at time x.

For both TPC-C and TPC-E, we find that the 2D graphs fall into two categories: A single device has the pattern of a diagonal line, while the rest of the devices all have similar patterns with a large number of scattered accesses. An explanation is that the device with the diagonal line pattern is the log device that sees mainly sequential writes, while OLTP data tables and indexes are stored on the other devices. Figure 2(a) and (b) show the patterns of the data devices in the two traces. Figure 2(c) shows the diagonal line pattern of the log device in TPC-E; the diagonal line pattern of the TPC-C log device is very similar. Since the log access pattern is easy to understand, we mainly analyze the I/O behaviors of the data devices in the following. We will study the log device at the end of the section.

Number of I/Os per Second. Figure 3 shows the number of I/Os for every second of a single data device in the TPC-C and TPC-E traces. For the steady portions of the traces, TPC-C sees an average 3360 I/Os per second, and TPC-E sees an average 2740 I/Os per second. The standard deviation of the I/O rate in the TPC-C (TPC-E) trace is 2.2% (3.6%) of the average. Thus, the two traces are roughly comparable.

I/O Request Breakdown. Table 2 shows the breakdown of device I/O request sizes and types in the two benchmark runs. The table reports the breakdown as a two-dimensional cube. The rows divide requests into three categories based on request sizes: 8KB, 16KB, or other. The columns divide requests into two categories based on request types: read or write. The table shows

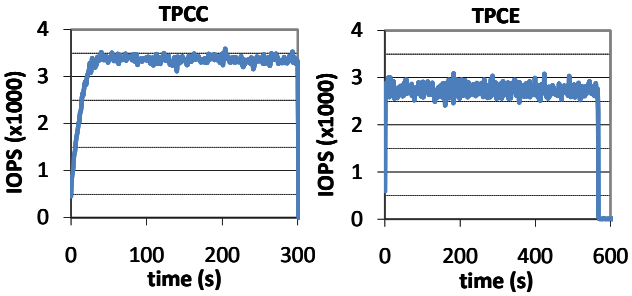


Figure 3: Number of I/Os for every second in the disk trace of a single data device.

Table 2: I/O request breakdown for data devices.

TPC-C			
Size	Read	Write	Both types
8KB	65.70%	32.66%	98.36%
16KB	0.00%	1.49%	1.49%
other sizes	0.00%	0.15%	0.15%
all sizes	65.71%	34.29%	100.00%

TPC-E			
Size	Read	Write	Both types
8KB	90.68%	8.29%	98.97%
16KB	0.00%	0.79%	0.79%
other sizes	0.01%	0.23%	0.24%
all sizes	90.69%	9.31%	100.00%

the percentage of I/O requests in each category as well as the one-dimensional and the total aggregate values.

From Table 2, we see that 8KB is by far the dominant size (98.36% in TPC-C and 98.97% in TPC-E). 16KB I/Os are the second most significant. I/Os of sizes other than 8KB are almost entirely writes. An explanation is that the DBMS's default I/O size is 8KB, while in some rare cases writes to contiguous disk blocks are merged together into a single request. This implies that the I/Os are quite random, which is reasonable in TPC-C. However, it seems counter-intuitive that TPC-E also has such behavior since it is expected to have more data skews. We will study this point further when analyzing the temporal and spatial locality of the benchmarks.

Focusing on the request type breakdown, we see that TPC-C sees 65.71% reads and 34.29% writes, a 1.9:1 read to write ratio, while TPC-E sees 90.69% reads and 9.31% writes, a 9.7:1 read to write ratio. Clearly, TPC-E is more read intensive than TPC-C. This observation

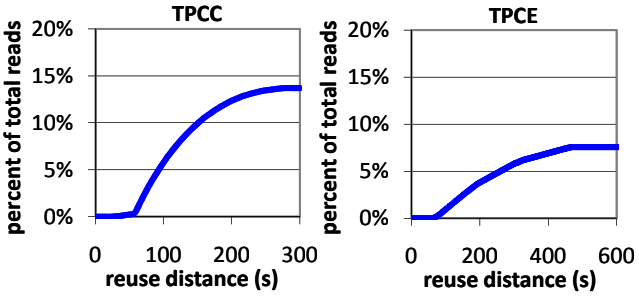


Figure 4: Cumulative distribution of read reuses. A read reuse is a read I/O to the same block address of a previous read I/O.

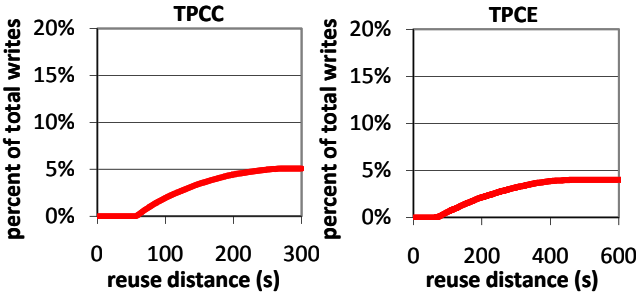


Figure 5: Cumulative distribution of write reuses. A write reuse is a write I/O to the same block address of a previous write I/O.

confirms Table 1; TPC-E is designed to have a higher percentage of read-only transactions than TPC-C.

Temporal Locality. We study the I/O temporal locality of the TPC-E and TPC-C benchmarks. We define a read reuse as an I/O read (R_2) to the same block address of a previous I/O read (R_1). The time from issuing R_1 to issuing R_2 is the reuse distance for R_2. If there are multiple previous I/O reads having the same block address as R_2, we use the most recent previous I/O read to compute the reuse distance of R_2. Similarly, we define write reuse and write reuse distance. In general, the shorter the reuse distance and the higher percentage of reuses in total I/Os, the more amenable the application is to I/O optimizations on temporal locality.

Figure 4 and Figure 5 show the cumulative distribution of read and write reuses for data devices in the TPC-C and TPC-E traces. The X-axis is reuse distance in seconds. The Y-axis is percent of total read (write) accesses. For read reuse, 100% is the total number of read accesses. For write reuse, 100% is the total number of write accesses. The curves are cumulative distributions. For example, in the TPC-E figures, the points at 600 seconds show that read reuses with reuse distance $\leq$ 600 seconds consist of 8% of all I/O reads, and write reuses with reuse distance $\leq$ 600 seconds consist of 4% of all I/O writes in the TPC-E run. Note that 600 (300) seconds is the trace length of TPC-E (TPC-C).

From Figure 4 and Figure 5, we see that TPC-E and TPC-C have quite similar curves: There is not much

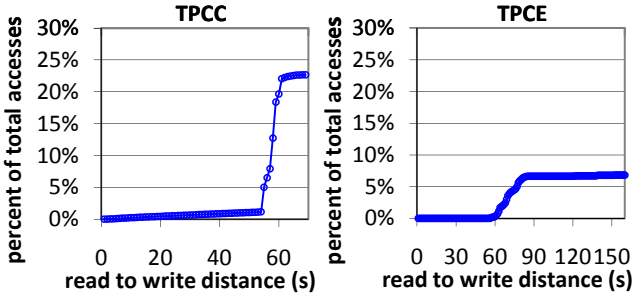


Figure 6: Cumulative distribution of write after read I/Os. A write after read is a write I/O such that there is a previous read I/O to the same block address but there is no other I/O to the same address in between.

temporal locality in either run. TPC-E has even fewer read reuses than TPC-C, which is counter-intuitive since TPC-E is designed to have more data skews. Moreover, both benchmarks start to see I/O reuses at around 60 seconds. This suggests that the main memory buffer pool may play a role in shaping the reuse patterns seen here. We study the buffer pool behavior next.

Understanding the Buffer Pool Behavior. We study the behavior of the buffer pool by looking at read to write distance in Figure 6. We define a write-after-read as a write I/O (W) to the same block address of a previous I/O read (R) and there is no other I/O to the same block address in between R and W. The read to write distance is the time from issuing R to issuing W. Typically, DBMSs perform updates by reading the destination database page into the main memory buffer pool (if it is not already in the buffer pool), and then making modifications to the page. Dirty page write backs are often done asynchronously by a background I/O cleaner thread/process. (Transaction durability is guaranteed by the synchronous redo log). The purpose of this strategy is to reuse pages in main memory for saving I/O operations. The read to write distance shows the duration that a page is kept in the memory buffer pool.

In Figure 6, the curves show the cumulative distributions of write after read I/Os. We see that there is a jump in both curves around 60 seconds. This means that the buffer pool keeps a page for about 60 seconds before writing it back. Note that this value is often determined by the DBMS configurations.

Explanation of the Temporal Locality Figures. Now we can use the understanding of the buffer pool behavior to explain the seemingly counter-intuitive observation in Figure 4 and Figure 5. First, the reuse distances are larger than 60 seconds because pages tend to stay in the main memory buffer pool for about 60 seconds, and any data reuses with distance less than 60 seconds would be captured inside main memory. Therefore, I/O devices do not see any reuses less than 60 seconds. Second, TPC-E has poor temporal locality because the data

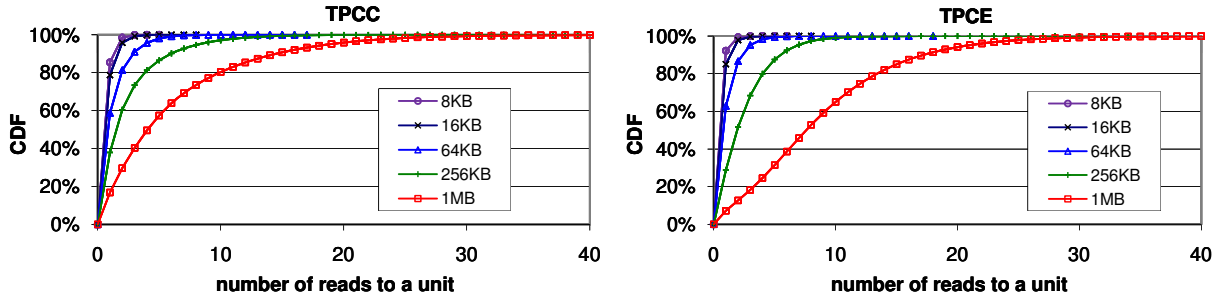


Figure 7: Cumulative distributions for number of read I/Os to a unit while varying unit size from 8KB to 1MB.

Table 3: 80-th and 99-th percentiles in Figure 7.

Unit Size	8KB blocks	TPC-E 80-th	TPC-E 99-th	TPC-C 80-th	TPC-C 99-th
8KB	1	1	2	1	2
16KB	2	1	3	2	3
64KB	8	2	5	2	6
256KB	32	4	10	4	14
1MB	128	8	29	10	28

skews in TPC-E may already be captured by the DBMS within the 60 second time when data pages are cached in the main memory buffer pool. Since the DBMS already effectively optimized away the data skews, the I/O devices see little temporal locality in the I/O accesses.

Spatial Locality. We would like to understand if I/O accesses tend to visit disk blocks that are near one another. As shown in Figure 7, we count the number of I/O reads for every unit where the unit size is a multiple of the default 8KB I/O size. We vary the unit size from 8KB to 1MB. For example, when the unit size is 1MB, we conceptually consider the I/O devices as organized into 1MB sized units. Then we count the number of I/O reads to every unit in the entire trace. We focus on units with non-zero counts. The 1MB curve shows the cumulative distribution of the per-unit count. Note that a steep curve means that most units see a small number of reads, indicating poor spatial locality.

Table 3 lists the 80-th and 99-th percentiles for Figure 7. For example, the table cell at the "TPC-E 99-th" column and the 1MB row means that 99% of 1MB-sized units see at most 29 I/O reads in the entire TPC-E trace. Note that a 1MB unit consists of 128 8KB blocks as shown in the second column in Table 3. Since the dominant I/O size is 8KB, this means that at most 29 out of the 128 8KB blocks are accessed in the entire trace. This shows very poor spatial locality at the 1MB granularity. Similarly, we see poor spatial locality at 256KB, 64KB, and 16KB granularities in both TPC-C and TPC-E. At the 16KB granularity, 85% of the units see a single access in TPC-E. Among the rest, 13% see 2 accesses, and 2% see 3 or more accesses. For comparison purpose, we also include the cumulative distribution of 8KB reads. 92% of 8KB units see a single access in TPC-E, while only 8% see more than one access, which corresponds to the read reuses in Figure 4.

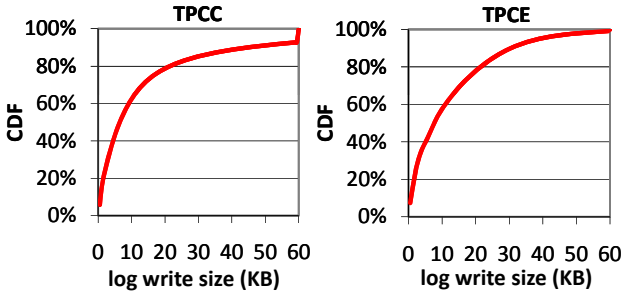


Figure 8: Cumulative distribution of log writes.

Log Write Size. For the most part, the sequential write behavior of the log device is well-known. Here, we are interested in the log write size distribution. Figure 8 shows the cumulative distribution of log writes. The TPC-C and the TPC-E curves are quite similar: About 80% of log writes are less than 20KB large. This confirms the previous observation that log writes are mainly small sequential writes [2]. Interestingly, the largest log writes in both TPC-C and TPC-E are 60KB (and TPC-C sees a non-trivial 7% log writes at 60KB). An explanation is that the DBMS logging manager has an upper limit of 60KB for flushing log writes from its log buffer.

Summary of Trace Study. In summary, we find in the trace study that (i) TPC-E is more read intensive than TPC-C, seeing a 9.7:1 read to write ratio; and (ii) both TPC-E and TPC-C traces display poor temporal and spatial locality. The former confirms TPC-E's design choice of a higher percentage of read-only transactions than TPC-C. However, the latter is seemingly counter-intuitive because TPC-E is designed to have more data skews than TPC-C. Our explanation is that the memory size (8% of the TPC-E database size) is large enough to capture the skewed accesses when the data are cached in the main memory buffer pool. As a result of this filtering effect, the I/O access pattern of TPC-E seen at storage devices is as random as TPC-C.

4. REPLAYING TRACES USING SSD

The trace study in Section 3 shows that TPC-E's I/O access pattern is as random as TPC-C. This implies that *like TPC-C, TPC-E can benefit from flash-based Solid State Drives (SSDs)*, which support much higher random I/O performance than HDDs. The main weakness

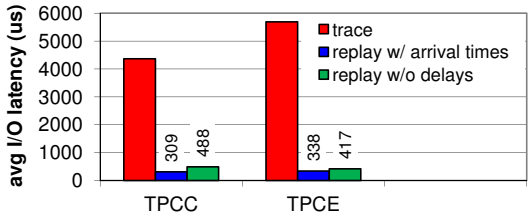


Figure 9: OLTP trace replay on an Intel X25-E SSD.

of SSDs is their relative high price/GB. Fortunately, device capacity is often a secondary issue in OLTP. For example, the TPC-C (TPC-E) trace utilizes only 4GB (5GB) capacity per HDD. Recent studies investigated the use of flash devices in OLTP systems [1, 2, 3, 5], but none of the studies examined TPC-E. Here, we verify the above understanding by comparing the performance benefits of SSDs for TPC-C and TPC-E.

We model an SSD-only target system. We replay the OLTP traces using a single 32GB Intel X25-E SSD on a Dell Precision 390 workstation equipped with a 2.66GHz Intel Core 2 CPU and 2GB DRAM running Linux 2.6.24. Since transactional logging can be effectively supported by flash devices [2], we focus on replaying traces for data accesses. We consider the following issues in the experimental setup:

- *Address mapping.* We map all the 1MB units with non-zero I/Os in the traces to a unified contiguous address space. We divide the unified address space into 32GB-sized chunks, each conceptually mapped to an SSD in the target system. The TPC-C and TPC-E traces require 50 and 56 SSDs, respectively. We replay the traces, one chunk at a time on the single SSD.
- *I/O access order and arrival times.* We keep the order of the I/O accesses in the traces during replay. For I/O arrival times, we use two different settings. In the first setting, we use the arrival times in the traces during replay. In the second setting, we replay the I/O accesses one after another without any delays.
- *SSD initial state.* Since the OLTP traces show random access patterns, we prepare the SSD by writing 32GB data using random 8KB writes before the replay.
- *Device write cache.* The I/O write latency in the traces is on average 0.6ms. This means that the writes are handled by the persistent caches in the RAID controllers. We model similar write optimizations by enabling the SSD write cache during the replay.

Figure 9 shows the average I/O latencies computed from the traces and measured in the SSD replay experiments. Compared to the computed latencies in the traces, we see that the average I/O latency is reduced by a factor of 14 and 17 for TPC-C and TPC-E when replaying the traces on the SSD with original I/O arrival times. For the no-delay replay, which can be regarded as a stress test, the SSD improves the average I/O latencies of the traces by a factor of 9 and 14.

Overall, we see similar dramatic improvements in both TPC-C and TPC-E by employing SSDs. This shows that because TPC-E sees I/O access patterns as random as TPC-C, techniques that improve random I/O performance in TPC-C, such as employing SSDs, can be effectively applied to optimize TPC-E performance.

5. CONCLUSION

In this paper, we compared the new TPC-E benchmark with the familiar TPC-C benchmark using two disk traces. We find that TPC-E is more read intensive than TPC-C: the read to write ratio is 9.7:1 in TPC-E vs. 1.9:1 in TPC-C. Moreover, we find that although TPC-E is designed to have realistic data skews, the skewed accesses can be well captured by the memory buffer pool. As a result, the TPC-E trace shows random I/O access patterns similar to TPC-C.

These findings have the following two implications. First, the higher read to write ratio in TPC-E means that I/O optimizations targeting writes may be less effective for TPC-E. Second, the random I/O access patterns in TPC-E imply that the conclusions of many previous I/O studies for TPC-C can be still valid. For example, we verified through an SSD replay study that like TPC-C, replacing the HDDs with SSDs can dramatically improve the I/O performance of TPC-E.

6. REFERENCES

[1] M. Canim, B. Bhattacharjee, G. A. Mihaila, C. A. Lang, and K. A. Ross. An object placement advisor for DB2 using solid state storage. *PVLDB*, 2(2):1318–1329, 2009.

[2] S. Chen. FlashLogging: exploiting flash devices for synchronous logging performance. In *SIGMOD*, 2009.

[3] G. Graefe. The five-minute rule twenty years later. In *DaMoN Workshop*, 2007.

[4] S. Kavalanekar, B. L. Worthington, Q. Zhang, and V. Sharda. Characterization of storage workload traces from production windows servers. In *4th International Symposium on Workload Characterization (IISWC)*, 2008.

[5] S.-W. Lee, B. Moon, C. Park, J.-M. Kim, and S.-W. Kim. A case for flash memory SSD in enterprise database applications. In *SIGMOD*, 2008.

[6] Transaction Processing Performance Council. TPC-C Benchmark Revision 5.10.1. http://www.tpc.org/tpcc/.

[7] Transaction Processing Performance Council. TPC-E Benchmark Version 1.9.0. http://www.tpc.org/tpce/.

[8] Transaction Processing Performance Council. TPC-E Benchmark Overview. http://www.tpc.org/tpce/spec/TPCEpresentation.ppt, February 2007.