Architecture and Performance Characteristics of a PostgreSQL Implementation of the TPC-E and TPC-V Workloads

Andrew Bond[1], Doug Johnson[2], Greg Kopczynski[3], and H. Reza Taheri[3]

[1] Red Hat, Inc.
[2] InfoSizing, Inc.
[3] VMware, Inc.
abond@redhat.com, doug@sizing.com, {gregwk,rtaheri}@vmware.com

Abstract. The TPC has been developing a publicly available, end-to-end benchmarking kit to run the new TPC-V benchmark, with the goal of measuring the performance of databases subjected to the variability and elasticity of load demands that are common in cloud environments. This kit is being developed completely from scratch in Java and C++ with PostgreSQL as the target database. Since the TPC-V workload is based on the mature TPC-E benchmark, the kit initially implements the TPC-E schema and transactions. In this paper, we will report on the status of the kit, describe the architectural details, and provide results from prototyping experiments at performance levels that are representative of enterprise-class databases. We are not aware of other PostgreSQL benchmarking results running at the levels we will describe in the paper. We will list the optimizations that were made to PostgreSQL parameters, to hardware/operating system/file system settings, and to the benchmarking code to maximize the performance of PostgreSQL, and saturate a large, 4-socket server.

Keywords: Database performance, virtualization, PostgreSQL, cloud computing.

1 Introduction

1.1 TPC-V Benchmark

In this paper, we will describe the architecture of the TPC-V benchmark, give a progress report on its implementation, and present the performance results collected so far. TPC-V measures the performance of a server running virtualized databases. It is similar to previous virtualization benchmarks in that it has many VMs running different workloads. It is also similar to previous TPC benchmarks in that it uses the schema and transactions of the TPC-E benchmark. But TPC-V is unique since unlike previous virtualization benchmarks, it has a database-centric workload, and models many properties of cloud servers, such as multiple VMs running at different load demand levels, and large fluctuations in the load level of each VM. Unlike previous TPC benchmarks, TPC-V will have a publicly-available, end-to-end benchmarking kit.

R. Nambiar and M. Poess (Eds.): TPCTC 2013, LNCS 8391, pp. 77–92, 2014.

We will start with a short introduction to virtualization, give a brief background on the properties and the development process of the benchmark, then describe the architecture of the kit, and conclude with some of the performance results obtained so far.

1.2 Virtualization

Virtualization on the Intel x86 architecture was pioneered in late 1990s [2, 3, 4], and has grown to become a mainstream technology used in enterprise datacenters. Today, virtualization is the fundamental technology that enables cloud computing. So, there is strong demand for a database-centric virtualization performance benchmark with cloud computing characteristics. In response to this demand, a TPC subcommittee was formed in 2010 to develop a benchmark with the following properties:

1. Models a database-centric workload
2. Exercises the virtualization layer
3. Has a moderate number of VMs (as opposed to modeling a pure consolidation scenario with a large number of VMs)
4. Emulates a mix of Transaction Processing and Decision Support workloads
5. A heterogeneous mix of low load volume and high load volume VMs
6. Has a healthy storage and networking I/O content
7. Models the elastic load-level variations of cloud VMs

The complete description of the benchmark specification, the details of the load variation, performance metrics, and other properties of the benchmark are detailed in [1, 5]. In this paper, we will describe the new developments and prototyping results.

2 Other Virtualization Benchmarks

2.1 Consolidation Benchmarks

The early virtualization benchmarks were representative of the consolidation environment where many low volume workloads that had been running on individual servers would be consolidated onto a single server using virtualization. The earliest example is VMmark [14] which is a de facto standard with hundreds of publication on several succeeding versions of the benchmark. An industry standard follow-on is SPECvirt_sc2010 [7] which incorporates modified versions of three SPEC workloads (SPECweb2005_Support, SPECjAppServer2004 and SPECmail2008) and drives them simultaneously to emulate virtualized server consolidation environments, much like VMmark 1.0 did. To date, there have been 33 publications on SPECvirt_sc2010. The SPECvirt_sc2013 [9] benchmark was released in 2013 with 2 publications so far.

2.2 TPC-VMS

In 2012, the TPC released the TPC-VMS [7] (TPC Virtual Measurement Single System) benchmark, which emulates a simple consolidation scenario of 3 identical databases. The 4 workloads used in TPC-VMS are the TPC-C [10], TPC-E [11], TPC-H [12], and TPC-DS [13] benchmarks. By leveraging existing TPC benchmarks,

TPC-VMS does not require development of a new kit. It is expected that the ease of benchmarking afforded by use of existing kits will result in vendors publishing TPC-VMS results while the more feature-rich TPC-V benchmark is being developed.

3 TPC-V Architecture

3.1 TPC-E as a Starting Point

We decided early on to base the TPC-V workload on the existing TPC-E [11] benchmark. The *long pole* in benchmark development is often the development of the schema and the transactions, as well as writing a crisp, detailed specification that lays out the detailed documentation required for audit and publication procedures. By borrowing the Data Definition Language (DDL) and Data Manipulation Language (DML) of TPC-E, we were able to start the prototyping of TPC-V much earlier than is typical of TPC benchmarks. And by using the TPC-E functional specification document as the starting point, we only had to focus on what is new in TPC-V. TPC-V is fundamentally a different benchmark from TPC-E with different characteristics, yet gained years of development time by using TPC-E as the foundation.

3.1.1 Differences with TPC-E

Like TPC-E, TPC-V has 33 tables and 12 transactions, and very similar DDL and DML. However, there are differences in table cardinalities and the transaction mix, mostly to make the benchmarks non-comparable and for ease of benchmarking [1].

3.1.2 VGen

EGen, a publicly available program, generates the raw data that is used to populate a TPC-E database. It is also linked with the benchmarking kit to produce the run time transaction parameters. This ensures that query arguments match what has been loaded into the database. It also governs the generation of many run-time parameters, such as the transaction mix frequencies and random numbers. Besides making it easier to develop benchmarking kits, this guarantees adherence to the benchmark specification

TPC-V follows this model by using a VGen module that is based on EGen, modified to conform to the TPC-V specification. As will be detailed in section 3, the TPC-V benchmarking kit must produce different volumes of load to different VMs (section 3.2), and vary this load at different phases of the benchmark run (section 3.4). We realized early on that driving the load to different VMs independently and attempting to keep them in sync at run time would be nearly impossible. Instead, all of these relationships are maintained by VGen. It distributes transactions over VMs following the numerical quantities specified in a configuration file, and also varies the load based on the elasticity parameters in that file. Using a deck of cards method, VGen ensures that the load ratios among the many VMs are maintained at the values specified in the configuration file. If one VM is running slower than expected, the load to other VMs is automatically reduced such that the specified ratios are maintained.

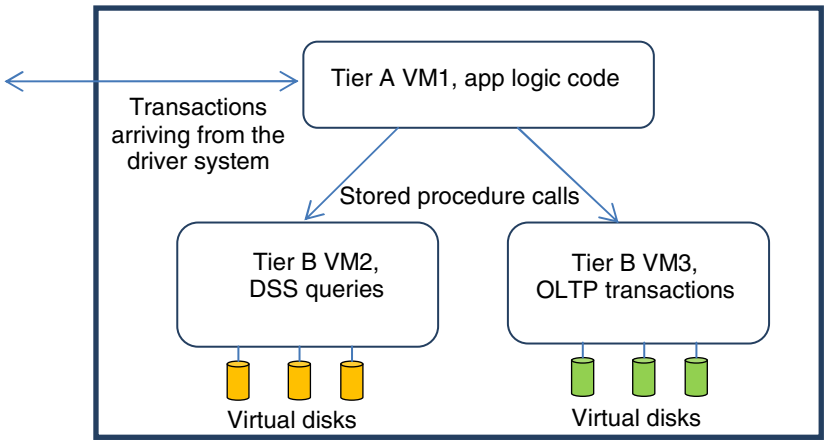

Fig. 1. Components of a TPC-V **Set**

3.2 Heterogeneous Load

The basic building block of TPC-V is a Set of 3 VMs. Tier A VM1 receives transactions from the driver system and runs the database client code, similar to the Tier A of a TPC-E benchmark configuration [11]. VM1 directs the two Decision Support transactions to the DSS VM2, and the other transactions to the OLTP VM3. Each VM has an independent database instance that resides on that VM's virtual disk drives.

3.3 Multiple Sets and Groups

Fig. 2 represents the simplest TPC-V configuration of a server with 4 Groups, each with one Set of 3 VMs for a total of 12 VMs. To emulate the heterogeneous nature of VMs in a cloud environment, each Group handles a different proportion of the overall load. Averaged over the full measurement interval, Groups A, B, C, and D receive 10%, 20%, 30%, and 40% of the overall load, respectively. The sizes of the independent databases in the 4 VMs (represented by table cardinalities) follow the same proportions. The 4 Groups are driven independently; the driver module is required to ensure that the load proportions remain as specified.

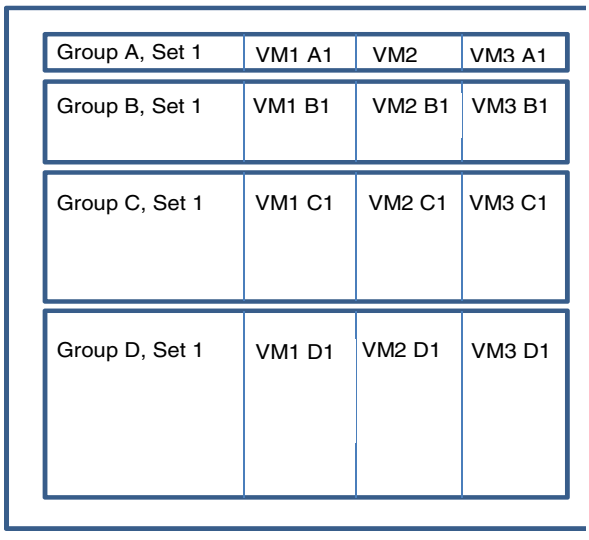

Fig. 2. A TPC-V server with 4 Groups and 12 VMs

The number of Sets per Group in TPC-V grows as the overall throughput grows. So, e.g., at a throughput level of 4,000 tpsV, the sponsor is required to configure 2 Sets per Group. For Group A, each of the two Sets supplies 5% of the overall throughput; a similar calculus applies to the other three Groups. The growth in the number of Sets per Group is sub-linear: a 10X throughput growth might result in a 2X increase in the number of Sets per Group. This is characteristic of database servers in the cloud.

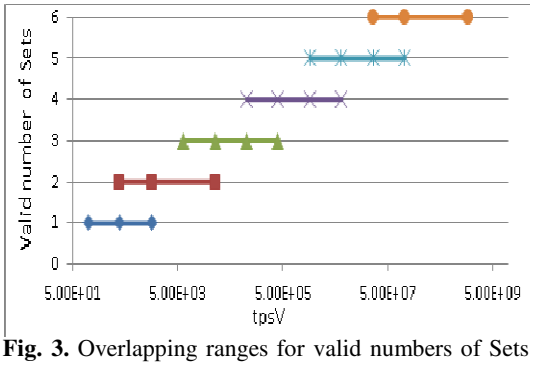

Fig. 3. Overlapping ranges for valid numbers of Sets per Group

Rather than requiring an exact number of Sets for every throughput value, we allow two possible Set counts for most throughput ranges, as shown in Table 1 and Fig. 3. This was done for ease of benchmarking. Without this allowance, if a test sponsor were targeting a throughput that is near the value at which the number of Sets per Group changes, a slight change up or down in the eventual throughput would necessitate rebuilding the testing infrastructure with a different number of VMs.

Table 1. Valid numbers of Sets for various throughputs

From tpsV	To tpsV	No. of **Sets**
100	1600	1
400	25,600	2
6,300	409,500	3
102,400	6,553,600	4
1,638,400	104,857,600	5
26,214,000	Infinity	6

So, for example, 25,600 tpsV is the crossing point from 2 to 3 Sets per group. If the sponsor expects to achieve 25,600 tpsV, builds a 3-Sets-per-Group configuration with 36 VMs and 24 databases, but reaches only 24,000 tpsV, there is no need to reconfigure platform with fewer VMs since the specification allows 3 Sets per Group down to 6,300 tpsV. The sponsor only needs to repopulate the databases, scaled to the correct throughput.

3.4 Elasticity

A feature of TPC-V is that the load of each Set rises and falls during the measurement interval. This represents the elastic nature of workloads present in cloud data centers, and the resource allocation policies required to handle such elasticity. The overall load presented to the System Under Test remains constant during the Measurement Interval, but the contribution from each Set varies by as much as a factor of 7X every 12 minutes, e.g., the rise of the contribution of Group A from 5% to 35% in Elasticity

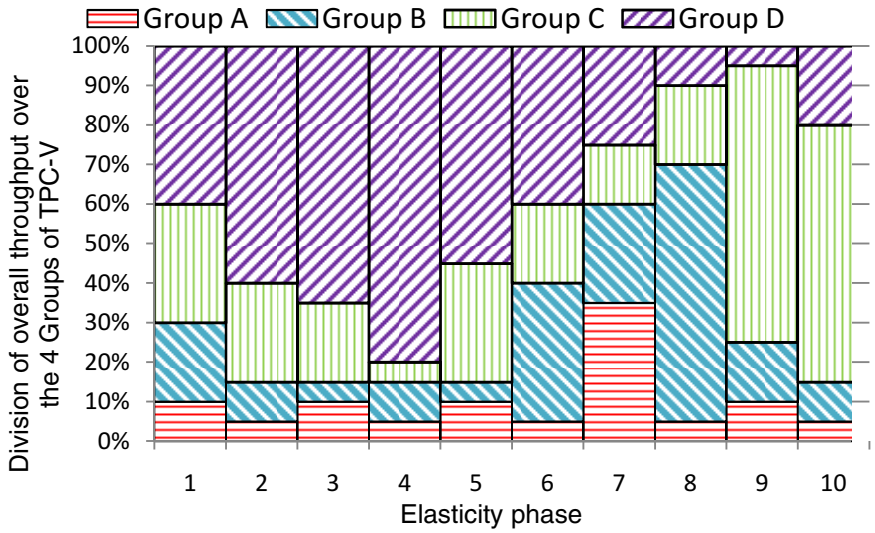

Fig. 4. Distribution of overall load over the 4 Groups versus time

Table 2. Phase-to-phase variation of load received by individual Groups

Elasticity Phase	Group A	Group B	Group C	Group D
1	10%	20%	30%	40%
2	5%	10%	25%	60%
3	10%	5%	20%	65%
4	5%	10%	5%	80%
5	10%	5%	30%	55%
6	5%	35%	20%	40%
7	35%	25%	15%	25%
8	5%	65%	20%	10%
9	10%	15%	70%	5%
10	5%	10%	65%	20%
Average	10%	20%	30%	40%

Phase 7. When the contribution of a Group changes, the contribution of all individual Sets in that Group change to the same degree. Table 2 and Fig. 4 show how much each Set contributes to the overall throughput in each 12-minute Elasticity Phase.

4 Reference Kit

Benchmarking kits for TPC benchmarks have always been provided by test sponsors, typically by DBMS vendors who tailor their kits to their own databases. Although we would have liked a DBMS vendor to provide a benchmarking kit for TPC-V, due to lack of such a commitment, the subcommittee accepted the challenge of developing its own kit. This turned out to be a positive development as it will result in the TPC releasing its first publicly available, complete end-to-end benchmarking kit which can be used by system vendors, researchers, and end users alike. The details of this decision making, comparison with other benchmarking kits, and a block diagram of the kit components can be found in [1]. Fig. 5 shows how the various elements of the TPC-V reference kit map to the components of the tested configurations.

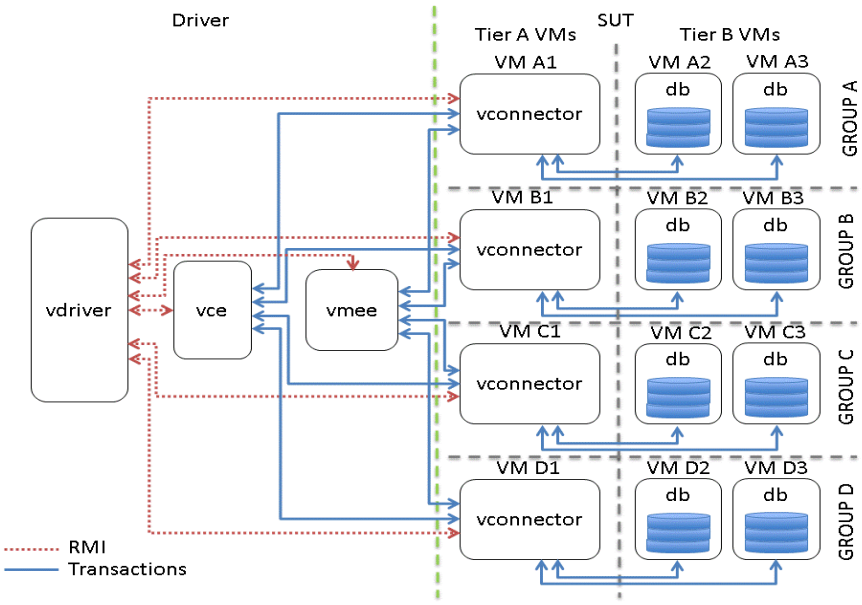

Fig. 5. Single-set Reference Driver Components Representation

4.1 V-Gen Functionality Development

The primary focus in implementing V-Gen functionality has been in adding multi-group, multi-set, multi-phase support. And while multi-group and multi-set and multi-phase have been described previously, the multi-iteration support has been added in order to be able to run as many ten-phase intervals as desired in a single test. The tester will then be able to choose any sequential ten phases in the multi-interval test run as the measurement set. The ability to choose such a measurement set is being added to a reporter process, which is also new to the kit. And lastly, the runtime result polling has been modified to provide group mix data that displays performance on a per-group basis in addition to the previous per-transaction basis.

4.2 Card Deck for Multi-group, Multi-set and Multi-phase Support

As described previously, multi-group and multi-set support has been implemented in the reference driver by having every CE process connect to each vconnector process in every group and every set. In doing so, we are able to use a card deck to assure the proper mix of transactions across these groups and sets. This deck is created for each CE load generating thread and is shuffled at the beginning of the run. Each time the CE starts a new request, it takes a card from this deck to determine the group and set ID of the vconnector process to whom it should direct the request, and once the bottom of the deck is reached, it simply starts back at the top.

Likewise, different phases have different transaction mixes, so we have a separate card deck for each of the ten phases that contains the proper mix of transactions for

that phase. At a phase change, the deck from which the cards are pulled is also changed to the corresponding deck with the correct request mix.

4.3 Result Reporting

A reporter class is under development to help with processing the mix logs. It is currently capable of combining CE mix logs from multiple CE processes into a merged log that can be used to extract the needed data for a benchmark report. One such piece of information that it currently offers is that after combining the CE mix logs, it creates a CSV file with the total number of transactions that occurred in each 30-second interval from the start of the ramp-up-phase to the end of the ramp-down phase of the full benchmark run. This code should require minimal modification to provide similar and more granular information on transaction totals over time based on transaction type, group, set, iteration, phase, or even per-client-thread transaction information.

Of course, to be able to accurately combine CE mix log files, you have to have information about the runtime configuration used to generate those logs. So at the end of a benchmark run, the reporter also creates a runtime.properties file that contains the necessary information. This file is also passed to the reporter when it is invoked.

4.4 Runtime Polling

The addition of groups and sets to TPC-V resulted in the need for group- and set-specific polling information. So in addition to the previous per-transaction-type

```
------------------------------------------------------------------------------
                    Txn Rate   Resp Time   Txn Pct   Pass Count   Fail Count
TRADE_ORDER      :      8.36      0.0077     11.33         1487           18
TRADE_RESULT     :      0.00           0      0.00            0            0
TRADE_LOOKUP     :      6.60      0.4995      8.95         1188            0
TRADE_UPDATE     :      1.66      0.6165      2.25          299            0
TRADE_STATUS     :     15.94      0.0074     21.60         2869            0
CUSTOMER_POSITION:     10.67      0.0076     14.46         1920            0
BROKER_VOLUME    :      4.09      0.0402      5.54          736            0
SECURITY_DETAIL  :     11.68      0.0067     15.83         2102            0
MARKET_FEED      :      0.00           0      0.00            0            0
MARKET_WATCH     :     14.78      0.0104     20.04         2661            0
DATA_MAINTENANCE :      0.00           0      0.00            0            0
TRADE_CLEANUP    :      0.00           0      0.00            0            0
------------------------------------------------------------------------------

------------------------------------------------------------
              Group 1     Group 2     Group 3     Group 4
Txn Total:        664        1329        3316        7971
Txn Pct  :       5.00       10.01       24.97       60.02
Resp Time:     0.0957      0.0771      0.0705      0.0624
Fail Cnt :          2           2           3          11
------------------------------------------------------------

Iteration 2 Phase 2 Aggregate Txn Rate: 73.78
```

Fig. 6. Sample polling output

polling information, per-group polling information has been added. So now a sample polling output might look like the output in Fig. 6. This additional information lets you know whether you are meeting the transaction mix requirements for each group, as well as the average response times and failure counts for each vconnector process.

4.5 MEE Development

As already noted, the MEE currently implements the Market Feed and Trade Result transactions as required for TPC-E. However, the nature of the MEE is such that it places constraints on implementation design for TPC-V. For example, we cannot design the MEE such that a single MEE process connects to all groups and sets as we do with the CE. This is because when transactions from the CE that trigger MEE transactions occur, they do not identify themselves by their group and set. Thus when the MEE generates a transaction in response to the CE trigger, it would have no way of knowing which vconnector process should be the recipient of this transaction.

Due to this design constraint, we need a MEE paired specifically with each vconnector process so that any CE request that triggers and MEE transaction will always be sent to the correct recipient. At this point, this could mean a separate MEE process is started for each vconnector process, but ideally we hope to be able to have one MEE process handle requests for all four groups in each set using separate transaction handling threads and requiring only unique connections for each of these four MEE threads. This is not a requirement, though having fewer processes for the prime client to coordinate with is certainly desirable.

4.6 TPC-E Functionality

Since TPC-E is the starting point of this benchmark, and since it is a simpler, single-system benchmark, we used it as the design center of the first implementation of the reference kit. Although a complete, compliant TPC-E kit is not a goal of this project, the early prototype has been used to provide a glimpse of PostgreSQL running the TPC-E workload. Although we have been experimenting with multiple Sets and VMs following the TPC-V architecture, the workload has been mostly based on TPC-E.

5 Current Status of the Benchmark and the Reference Kit

The TPC-V reference benchmarking kit is nearly complete as of this writing. Below are the functionalities that are completed:

- A Driver module that generates TPC-E or TPC-V transactions, and distributes them over any number of Set and Groups of VMs in case of TPC-V (see section 3). It also implements the TPC-V elasticity feature
- A VGen module based on the TPC-V schema, transaction mix, etc.
- The Customer Emulator module
- The Market Exchange Emulator module for TPC-E transactions
- The vconnector module that performs all the database accesses

- The DDL and DML scripts for PostgreSQL 9.2
- Linux shell scripts to launch all these programs, collect data and statistics, and produce results metrics

The functionalities that remain to be completed are:

- Modifying the MEE, stored procedures, and DML calls such that the Trade-Result and Market-Feed transactions conform to the TPC-V specification
- The Data-maintenance transaction (a non-critical component)
- Extensive prototyping results for verification and testing of the reference kit
- Porting of the reference kit to multiple environments

6 Results from Prototyping Experiments

6.1 Introduction

Most of the results presented here were obtained before the MEE functionality was added to the kit. So they are not an accurate representation of eventual TPC-V performance. However, we expect the two missing transactions to have similar profiles to the 8 transactions implemented. The current functionality is sufficient to study how efficiently PostgreSQL executes the TPC-E/TPC-V queries, as well as an analysis of whether the hypervisor used in the study was able to handle the variability and elasticity of the load that TPC-V places on the system. For the remainder of this section, we will refer to transactions per second or tps to denote the total number of transactions processed. This should not be confused with the tpsV metric, which only counts the Trade-Result transactions, which make up only a 10% fraction of the total transaction volume. Trade-Result is issued by the MEE module, which was not developed in time for our initial measurements. Hence we count all 8 transactions, and report that as tps.

6.2 Benchmarking Configuration

The system under test was a 4-socket HP ProLiant DL580 G7 server with 2.40GHz Intel Xeon E7-4870 (WestmereEX) CPUs. To put this in perspective, HP has published a TPC-E result of 2,454 tpsE[1] on this system. The highest TPC-E result is 5,457 tpsE on an IBM System X3850 X5 server[2]. So the server we are using for prototyping is a large, high-end server. The storage was two EMC VNX5700 disk arrays. 38 EFDs (EMC term for SSDs) in a RAID5 configurations were used for the DSS VMs, which have the lion's share of disk I/O. 88 spinning disk drives in a RAID 1 configuration were used for the OLTP VMs, which have lower I/O requirements. The software stack was vSphere 5.1, RHEL 6.1, PostgreSQL 9.2.2 and unixODBC 2.2.14.

The benchmark was configured with 1 Set for each of the 4 Groups, for a total of 12 VMs. The driver system was the 13th VMs on the system. The database size is expressed in Load Units, each LU representing 1,000 rows in the Customers table. The cardinalities of the other 32 tables are either fixed, or are proportional to the number of Customers.

---

1 As of 6/21/2013. Complete details available at `http://www.tpc.org/4046`

2 As of 6/21/2013. Complete details available at `http://www.tpc.org/4063`

Table 3. Configuration info for VMs

	VM A1	VM A2	VM A3	VM B1	VM B2	VM B3	VM C1	VM C2	VM C3	VM D1	VM D2	VM D3
DB size in LUs	-	50	50	-	100	100	-	150	150	-	200	200
DB size in GB	-	336	328	-	670	654	-	1004	980	-	1308	1328
Memory in GB	2	88	39	2	146	54	2	220	68	2	278	78
vCPUs	3	4	12	5	8	24	6	12	30	8	16	40

Table 3 shows various configuration parameters for the 12 VMs. VM1s have very little memory usage, and their CPU usage is about 1/8th of the total CPU load. VM2s have modest processing needs, but we had to allocate most of the memory to them to cache more of the database and reduce the I/O load. VM3s didn't need as much memory since their I/O was already low, but were allocated about 60% of the total processing power. It is worth noting that, much like real cloud database VMs, although the CPU resources were *overcommitted* (more virtual CPUs in the VMs than physical CPUs on the server), the total memory allocated to the 12 VMs is 979GB, on a server with 1TB of memory. This is common for database VMs since overcommitting memory can result in paging, with disastrous results for database performance.

Virtual CPUs and Elasticity

The number of virtual CPUs, however, totals 168, well above the 80 logical CPUs (40 cores X 2 hyperthreads per core) on the server. This overcommitting is common in cloud environments since the number of virtual CPUs configured into a VM should be adequate for its peak demand. But not all VMs peak at the same time. So as long as the total load does not exceed 80 CPUs' worth, we can overcommit the virtual CPUs.

6.3 1-Phase and 10-Phase Runs

As mentioned in section 3.4, TPC-V requires the load received by each Group to vary over ten 12-minute elasticity phases. As we will see in section 6.4, this posed a challenge in our environment due to storage bandwidth limitations. So we ran some experiments with a single phase (i.e., constant proportioning of load across Sets for the duration of the run) to study the performance characteristics of the database, the operating, the hypervisor, and the hardware. We also ran experiments with 10 phases to specifically study the ability of the system under test to respond to load elasticity, and to determine whether the TPC-V benchmarking kit is able to deterministically distribute the load over the Sets even when some Sets are strained under the load.

In a 1-phase run with 4 Groups, the throughput was 4,191 transactions per second[3]. In the CPU utilization graphs in Fig. 7, the Y axis is the total CPU utilization of each

[3] As mentioned in section 6.1, this *transactions per second* metric should not be confused with the tpsV metric, which would have been as much as an order of magnitude smaller.

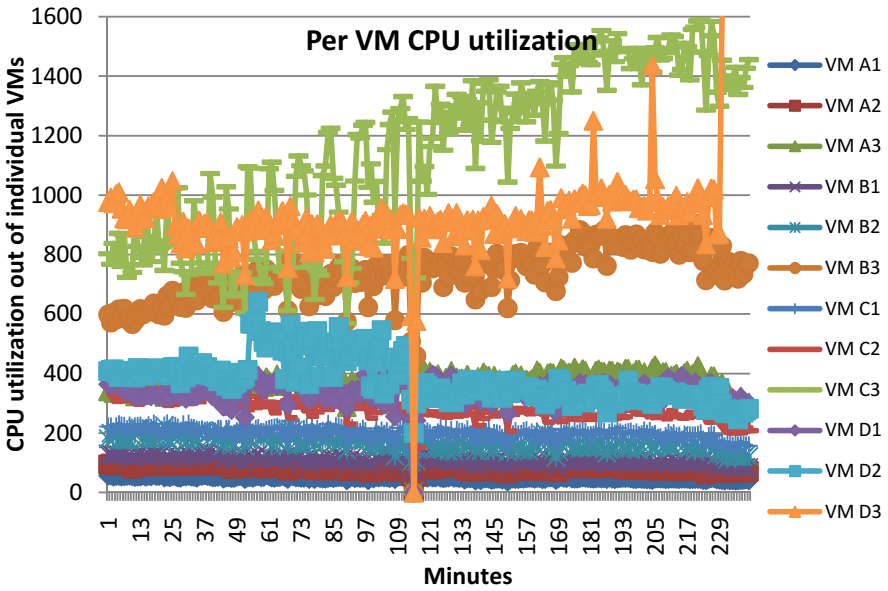

Fig. 7. CPU utilization of individual VMs for a single-phase run

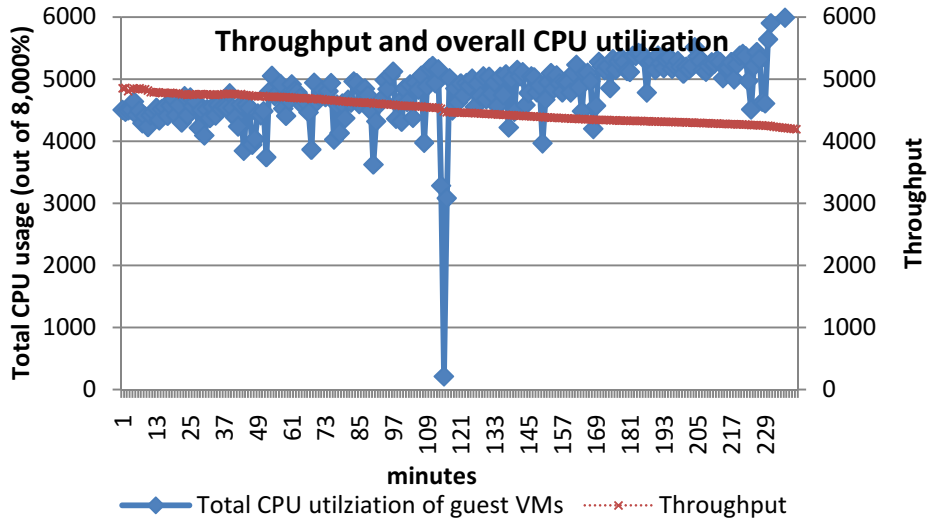

Fig. 8. Total CPU usage and throughput of a single-phase run

VM. An 8-vCPU VM would register a utilization of 800% if all 8 vCPUs were fully utilized. All of these metrics are measured on virtual CPUs on the guest VMs.

Fig. 8 shows throughput and the sum of CPU utilizations of individual VMs. It might appear that the system is not fully saturated, but that's due to the artifacts of hyperthreading when we collect statistics on the guest OS. Hypervisor and hardware counters register between 85% and 95% utilization on the CPU cores.

6.4 Throughput versus Other Performance Metrics for 10-Phase Runs

We also ran the benchmark with the load variation depicted in Fig. 4. As Fig. 9 shows, the CPU utilizations of individual VMs varied during the 2-hour runs, as did the overall throughput, shown in Fig. 10. However, the benchmarking kit ensured that the contributions of each Group remained exactly as prescribed in Table 2.

In this case, the throughput dropped drastically during some phases. The reason for this drop was the inability of the storage to cope with the changes in load. Briefly, the

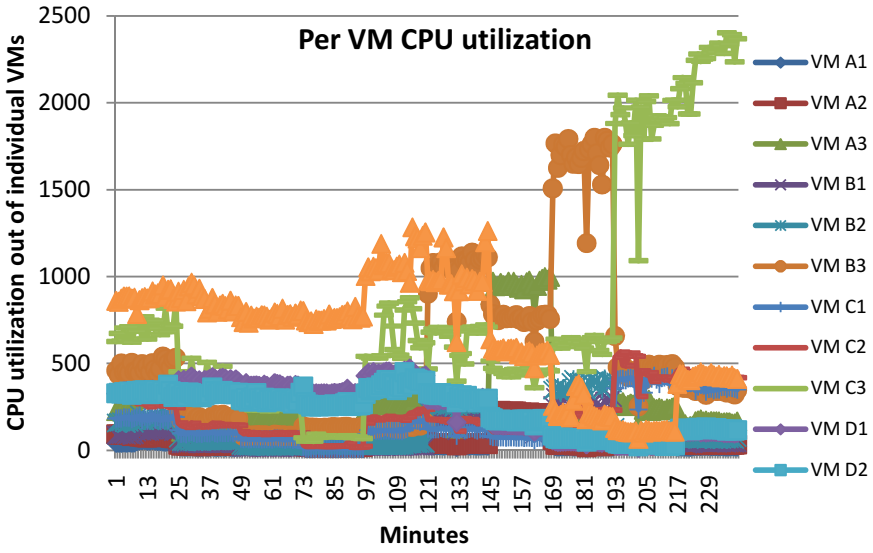

Fig. 9. CPU utilizations of individual VMs for a run with 10 elasticity phases

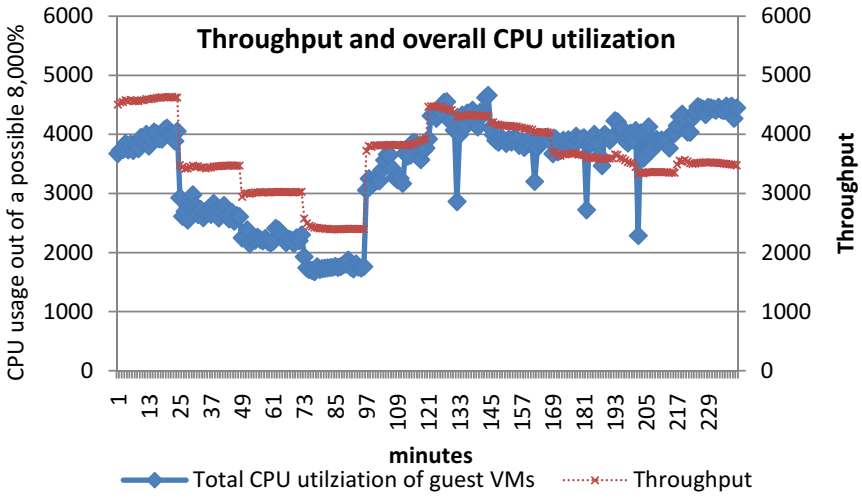

Fig. 10. Total CPU usage and throughput for a run with 10 elasticity phases

overall load, and hence the overall I/O requirements, remain constant over the execution time. Hence, if the storage is shared by all VMs in a striped format, the variations in load should not have the large impact that we observed. However, our storage was split in two groups: LUNs for Groups A and B were striped across one set of SSD disks, and LUNs for Groups C and D on a second set of SSD drives.

When in Phase 4 the load of the Group D is at its maximum, the second storage array was unable to satisfy the needs of that Group. One can overlay Fig. 4 and Fig. 10, and see that whenever Group C or Group D is at or near peak contribution to the overall throughput, performance goes down because we are unable to utilize excess capacity left in the first storage array dedicated to Groups A and B. In other words the benchmark is working exactly as intended: it is exposing a problem in the resource management of the underlying platform.

6.5 Results with a Full, End-to-End Kit

As pointed out in section 6.1, most of the results reported here were from a kit that did not have the MEE module, i.e., it was missing the important Trade-Result and Market-Feed transactions. In the months leading to this publication, we were able to take runs with a functional MEE, and could measure performance with the full complement of the 10 transactions (the Data-Maintenance transaction, which does not impact performance, has not been implemented). As we had predicted, the overall performance in terms of average milliseconds/transaction and the overall execution profile did not change very much. The addition of the two new transactions only changed the frequency percentages of the mix of transactions.

Early results look encouraging. We took runs with the TPC-E workload on a 16-way VM on the server described in section 6.2. We observed a throughput of roughly 140 tpsE at 80% CPU utilization on a 16-vCPU VM. So we are at ~9.1 milliseconds/tpsE. The published result with a commercial DBMS for this 80-way server is 2,454 tpsE, i.e. ~3.3 milliseconds/tpsE. Since our results are on a VM, there is a virtualization overhead of roughly 10% to consider. Also, our database was oversized, and our I/O rate is as much as 8 times the I/O rate of the commercial database due to PostgreSQL not having the Clustered Index feature of the commercial database. Considering all this, and assuming we can compare 16-way and 80-way results, performance is respectable for this early stage of prototyping.

6.6 PostgreSQL Tuning

Our current throughput level is close to 5,000 tps, summed over 4 Sets with 12 VMs. The audited result for this system is 2,454 tpsE, which only counts Trade-Result transactions. So running TPC-E with a commercial DBMS, it really processes 24,545 transactions per second. So we are nearly 5X off that mark. To make a direct comparison, we need to run a single VM with the complete TPC-E workload, including the 2 MEE transactions. But based on the data collected so far, we can see that many tuning opportunities exist, especially in the I/O rate. It appears that due to not having clustered indexes, PostgreSQL issues nearly 4 times as many I/Os per transaction as the TPC-E design goal. This is our primary focus area for the next phase of this project.

Table 4. I/O stats for DSS VM with one and two 2 file

		wrqm/s	r/s	w/s	rkB/s	wkB/s	avgrq	avgqu	await
1 FS	Data+log	1830	11151	2767	138602	33956	25	30	2.14
2 FS	data	2406	12350	2278	181902	18737	27	40	2.71
	log	343	0.34	134	1	17854	264	0.3	1.87

Table 5. I/O stats for OLTP VM with one and two 2 file systems

		wrqm/s	r/s	w/s	rkB/s	wkB/s	avgrq	avgqu	await
1 FS	Data+log	403	542	476	7682	5552	27	5.1	4.75
2 FS	Data	194	860	145	15613	1357	34	6.3	6.29
	log	1	0.04	225	0.16	3066	27	0.3	1.15

File System Parameters

An optimization recommend in [6] is separate file systems for data and Write-Ahead Log (WAL), because of the more strict cache flushing semantics for the log. Initially, an ext4 file system held both log and data, mounted with `noatime,nodiratime, nobarrier`,. We then created a pg-xlog ext3 file system, mounted with `noatime, nodiratime,data=writeback`. The log virtual disks of all VMs were placed on a LUN with only 4 disk drives, yet all experienced fast disk latencies. The result was a 6.5% increase in the throughput of the 4-Group, single-phase runs to 4,769 tps.

Checkpointing

Two parameters manage the checkpoint frequency of PostgreSQL. A new checkpoint is initiated either when a checkpoint has not occurred in `checkpoint_timeout` minutes, or when `checkpoint_segments` 16MB WAL segments have been used since the last checkpoint. We increased `checkpoint_timeout` from the default of 3 minutes to 30, and `checkpoint_segments` from the default of 3 to 128, believing 128 `checkpoint_segments` were enough, even for the largest VM, to let checkpointing be governed by `checkpoint_timeout`. Tests, however, showed that we were checkpointing as often as once every 2 minutes. We needed to increase `checkpoint_segments` to 1,920 segments on the largest VM; we used 5,120 to be safe. This change gave us a 2% improvement to 4,841 tps. Table 6 has the *background writer stats* section of the pgstatspack outputs before and after the change for 30-minute runs. The checkpoints_timed and checkpoints_req counts show that originally, there were 15 checkpoints triggered because the database had used all the WAL segments, and none due to reaching the checkpoint frequency timer. After increasing the number of WAL segments, we see only a single time-triggered checkpoint.

Table 6. Effects of increasing WAL_segments

Checkpoint metric	12 segments	5,120 segments
checkpoints_timed	0	1
checkpoints_req	15	0
buffers_checkpoint	4,437,177	956,174
buffers_clean	14,069	852,893
buffers_backend	46,297	39,297
buffers_alloc	24,831,473	23,749,499

7 Conclusions

The TPC-V reference benchmarking kit, which is at the heart of the benchmark, is nearly complete. It provides all the novel properties of TPC-V: a heterogeneous combination of workloads driven to many VMs, a deterministic distribution of load over the VMs regardless of how each VM handles the load, and dynamically varying the load levels to VMs to emulate the elasticity of load in the cloud. Using this kit, we have discovered several optimizations for a PostgreSQL implementation of TPC-V.

Acknowledgements. We thank Cecil Reames for VGen and specification reviews, Matt Emmerton, John Fowler, and Jamie Reding for TPC-E knowledge, Karl Huppler and Wayne Smith for high level benchmark requirements, and Jignesh Shah for PostgreSQL advice.

References

1. Bond, A., Kopczynski, G., Reza Taheri, H.: Two Firsts for the TPC: A Benchmark to Characterize Databases Virtualized in the Cloud, and a Publicly-Available, Complete End-to-End Reference Kit. In: Nambiar, R., Poess, M. (eds.) TPCTC 2012. LNCS, vol. 7755, pp. 34–50. Springer, Heidelberg (2013)
2. Figueiredo, R., Dinda, P.A., Fortes, J.A.B.: 'Guest Editors' Introduction: Resource Virtualization Renaissance. Computer 38(5), 28–31 (2005), http://www2.computer.org/portal/web/csdl/doi/10.1109/MC.2005.159
3. Nanda, S., Chiueh, T.-C.: A Survey on Virtualization Technologies. Technical Report ECSL-TR-179, SUNY at Stony Brook (February 2005), http://www.ecsl.cs.sunysb.edu/tr/TR179.pdf
4. Rosenblum, M., Garfinkel, T.: Virtual Machine Monitors: Current Technology and Future Trends. Computer 38(5), 39–47 (2005)
5. Sethuraman, P., Reza Taheri, H.: TPC-V: A Benchmark for Evaluating the Performance of Database Applications in Virtual Environments. In: Nambiar, R., Poess, M. (eds.) TPCTC 2010. LNCS, vol. 6417, pp. 121–135. Springer, Heidelberg (2011)
6. Smith, G.: PostgreSQL 9.0 High Performance. Packt Publishing (October 20, 2010)
7. Smith, W.D., Sebastian, S.: Virtualization Performance Insights from TPC-VMS, http://www.tpc.org/tpcvms/tpc-vms-2013-1.0.pdf
8. SPECvirt_sc2010 benchmark info, SPEC Virtualization Committee, http://www.spec.org/virt_sc2010/
9. SPECvirt_sc2013 benchmark info, SPEC Virtualization Committee, http://www.spec.org/virt_sc2013/
10. TPC: Detailed TPC-C description, http://www.tpc.org/tpcc/detail.asp
11. TPC: Detailed TPC-E Description, http://www.tpc.org/tpce/spec/TPCEDetailed.doc
12. TPC: TPC Benchmark H Specification, http://www.tpc.org/tpch/spec/tpch2.14.4.pdf
13. TPC: TPC Benchmark DS Specification, http://www.tpc.org/tpcds/spec/tpcds_1.1.0.pdf
14. VMware, Inc., http://www.vmware.com/products/vmmark/overview.html