Profiling the performance of virtualized databases with the TPCx-V benchmark

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 proliferation of virtualized servers in data centers has conquered the last frontier of bare-iron servers: back-end databases. The multi-tenancy issues of elasticity, capacity planning, and load variation in cloud data centers now coincide with the heavy demands of database workloads; which in turn creates a call for a benchmark specifically intended for this environment.

The TPC-V benchmark will fill this need with a publicly-available, end-to-end benchmark kit. Using a prototype of the kit, we profiled the performance of a server running 60 virtual machines with 48 databases of different sizes, load levels, and workloads. We will show that virtualized servers can indeed handle the elasticity and multi-tenancy requirements of the cloud, but only after careful tuning of the system configuration to avoid bottlenecks.

In this paper, we will provide a brief description of the benchmark, discuss the results and the conclusions drawn from the experiments, and propose future directions for analyzing the performance of cloud data centers by augmenting the capabilities of the TPCx-V benchmark kit.

Keywords: Database performance; virtualization; SQL Server; workload consolidation; performance tuning; cloud computing

1 Introduction

Server virtualization is ubiquitous in data centers, whether in the cloud or on users' premises. 32% of the new servers shipped in 2014 were deployed as virtualized servers [13]. In the early days of virtualization, database applications were deemed too demanding to be virtualized, but today's virtualized servers routinely run database applications.

Benchmarking database applications has always been a challenge due to the complexity and demanding nature of the applications. Virtualization adds an additional level of complexity, making it harder to both design and use such a benchmark. The Transaction Processing Performance Council (TPC) has been working on developing the TPCx-V benchmark to fill this need.

Prior publications [1, 2, 8] have reported on the TPCx-V's design philosophy, detailed architecture, and specific properties. We will not repeat those details in this paper.

adfa, p. 1, 2011.

Instead, we will provide a status update on the more recent changes, and show how the benchmark was used to measure and optimize performance on a large server.

2 Other virtualization benchmarks.

Prior to TPCx-V, there have been 3 other industry-standard, virtualization-specific benchmarks: VMmark 2.x, SPECvirt_sc2013, and TPC-VMS.

The earliest virtualization-specific benchmark was VMware's VMmark [18]. In its latest version, VMmark 2.x has evolved into a multi-host data center virtualization benchmark that includes both application-level workloads and platform-level operations, such as guest VM deployment, dynamic virtual machine relocation (vMotion) and dynamic datastore relocation (storage vMotion). VMmark has a *tile* architecture. Each tile includes 6 workloads of set workload levels. The user is expected to keep adding tiles until the system reaches peak throughput.

SPEC's SPECvirt_sc2013 [14] is another server consolidation benchmark with a tile-based architecture. Each tile includes workloads from earlier SPEC benchmarks SPECweb2005, SPECjAppServer2004, SPECmail2008 and SPECINT2006.

Neither VMmark 2.x nor SPECvirt_sc2013 addressed database workloads, and had lightly-loaded VMs with little storage I/O demands. The first virtualization benchmark with a database workload was TPC's TPC-VMS benchmark [3]. Although TPC-VMS was adequate in emulating a simple server consolidation scenario, its shortcomings included having a single DBMS workload, a constant count of 3 VMs, and no variation in the level of the loads of VMs.

3 TPCx-V benchmark

A TPC Subcommittee has been working on the development of the TPCx-V benchmark since 2010. The benchmark specification and the Express benchmark kit are nearly complete, and the development subcommittee is planning to submit the benchmark to the TPC General Council for final review and approval in August or November 2015.

3.1 Genesis of TPCx-V

The goal of TPCx-V is to measure how a virtualized server runs database workloads. It uses a database workload to measure the performance of virtualized platforms, notably the hypervisor, the server hardware, storage, and networking. To save development time, it relies on a prior TPC benchmark, in the same manner that SPECvirt_sc2013 used prior SPEC benchmarks as its workloads. The goal for TPCx-V was not to introduce a new database workload. The Subcommittee started out with the TPC-E [15] benchmark as the foundational workload for TPCx-V. However, the TPCx-V workload has evolved to be different from TPC-E in many ways. So comparing TPC-E results and TPCx-V results would be erroneous, as well as against the TPC policies.

Consult [1, 2] for details of the TPCx-V architecture. The full functional specification of TPCx-V will be available when the benchmark is officially released.

3.2 TPCx-V properties

The original design goals of TPCx-V were:

- Simulate cloud computing with:
  - A mix of On Line Transaction Processing (*OLTP*) and Decision Support Systems (*DSS*) workloads
  - Use databases of different sizes and load levels
  - Vary load levels to each VM to represent the elastic nature of load levels on cloud computing servers
- Devise a workload that stresses the virtualization layer and drives the state of the art for future hypervisor designs
- A Tiled architecture that requires more Tiles on larger servers
- But unlike earlier virtualization benchmarks, the load of TPCx-V Tiles is not constant: as in real world, larger servers run larger VMs, not just more VMs
- Improved ease of benchmarking compared to TPC-E. For example, the TPC-E schema makes it impossible to initially populate the database for one performance level, but run against a subset of the loaded data. TPCx-V schema has been updated to allow a benchmark sponsor to initially populate L_1 Load Units[1], but run against L_2 Load Unit, $L_2 < L_1$.
- Currently, the TPCx V kit is written to run on PostgreSQL. Future kit revisions may add the ability to use other databases

3.2.1 Performance Metric

TPCx-V has a predefined mix of transactions that are used to simulate the business activity of processing a trade. The Trade-Result transactions make up 10% of this mix. The Performance Metric reported by TPCx-V is ***tpsV***, which is a "business throughput" measure of the number of completed Trade-Result transactions per second.

3.3 TPCx-V architecture

3.3.1 Tiles, Groups, and VMs

The System Under Test (*SUT*) is divided into multiple Tiles. ***Tile*** is the unit of replication of TPCx-V configuration and load distribution. Each Tile consists of 4 ***Groups***. A valid TPCx-V configuration may have between 1 and 6 Tiles, with all Tiles contributing identical proportions of the total load. The number of Tiles and the number of Load

---

[1] A Load Unit represents 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.

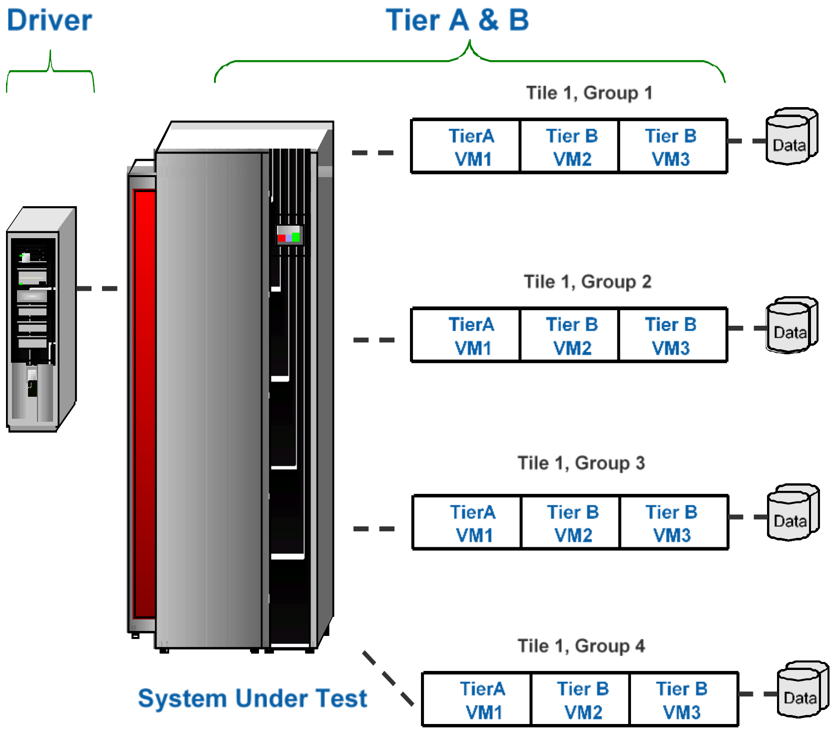

Fig. 1. One Tile, 4 Groups, and 12 VMs in a simple TPCx-V configuration

Units configured in the initial populations of the databases in each Group depend on the throughput, and are determined by a formula defined in the TPCx-V specification.

Each Tile has four ***Groups***, with Groups 1, 2, 3, and 4 contributing an *average* of 10%, 20%, 30%, and 40% of the total throughput of the Tile, respectively.

Each Group consists of one ***Tier A Virtual Machine*** and two transaction-specific ***Tier B Virtual Machines***. So there are a total of 12 VMs in each Tile as seen in **Fig. 1**.

VM1 of each Group contains that Group's Tier A, which runs the business logic application, and has the *frames* code functions that issue the database transactions. VM1 does not contain a database. VM2 is the Tier B VM that holds the DSS database, and accepts the 2 storage load-heavy DSS transactions. VM3 is the Tier B VM that holds the OLTP database, and accepts the 9 CPU load-heavy OLTP transactions.

3.3.2 Elasticity

Each of the 4 Groups in a Tile contributes a different amount of that Tile's overall load. Although the total load offered by a Tile remains constant over the 10 12-minute ***Phases*** of a benchmark run, the distribution of that load over the 4 Groups varies greatly, as

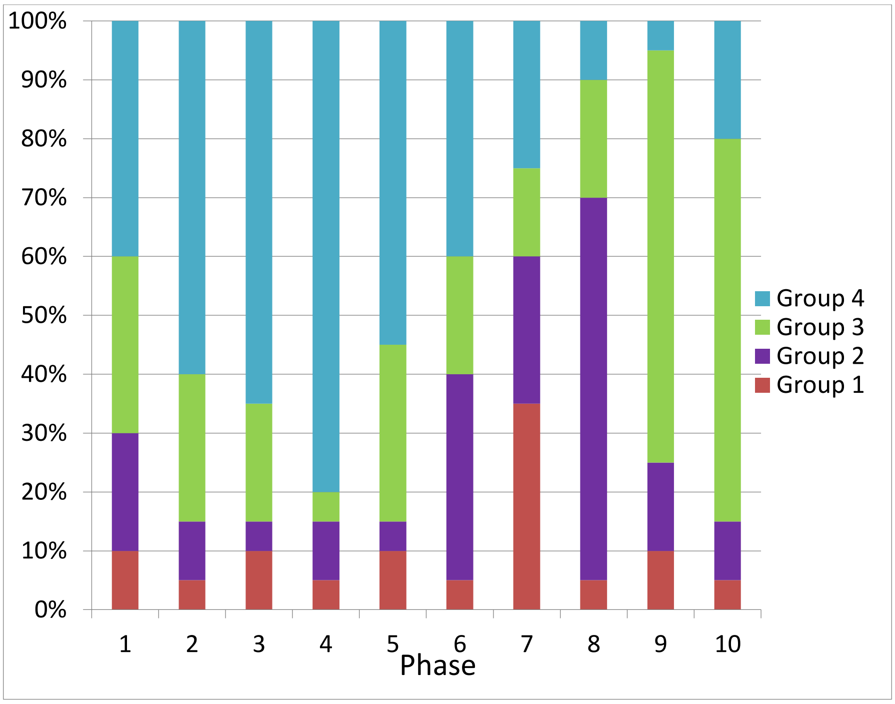

Fig. 2. TPCx-V dynamic load variation

depicted in **Fig. 2**. This is done to better emulate the elasticity of the load offered by different tenants of a server in a data center in the cloud.

3.3.3 Ensuring the balance of load over Tiles and Groups

A novel property of the TPCx-V benchmark kit is ensuring that the relative ratios of the loads offered to the Tiles and Groups conform to the specification requirements. Benchmarks typically vary the number of threads of execution in the benchmark driver to match the load level on the SUT. For example, if there are three Tiles, 1/3 of the driver threads would direct transactions to each of the three Tiles. The fundamental problem with this approach is that if one of the Tiles is too slow, it will fall behind the other two, and we won't have equal performance levels. Rebalancing the load offered to the Tiles is a complex, error-prone task. The TPCx-V benchmark avoids that complexity by having each driver thread distribute its transactions over all Tiles and Groups according to the specification requirements, implemented via a deck of cards algorithm. If three Tiles should receive equal loads, each thread uses a deck of cards with equal numbers of cards for all Tiles. If one Tile slows down, the driver thread will automatically issue transactions more slowly to ***all*** Tiles. Similarly, a deck of cards method is used to ensure each Group in a Tile receives the proper portion of that Tile's load. The Groups deck is changed at every Phase change. As a result, the benchmark kit is faithful to the ratios specified in the test configuration file to a very high level of precision.

4 TPC Express benchmarks

The TPC has operated for many years with the same benchmark development and results submission process (TPC Enterprise). TPC Express represents a shift in aspects of the development process and the benchmark execution process. A central component of this shift is the TPC-provided benchmark kit.

4.1 Role of the benchmarking kit in the Express benchmark

In the TPC Enterprise model, the TPC would develop a benchmark specification and it was up to the test sponsor to develop a compliant implementation. This is a non-trivial task. It requires expertise in a variety of areas including software development and performance tuning and optimization. Additionally, it requires a deep understanding of the benchmark specification and the complex subtleties of its many constraints. The net effect can be a prohibitively high bar for otherwise-would-be test sponsors.

TPC Express looks to minimize the cost of entry by utilizing a TPC-provided benchmark kit. With a TPC-provided kit there is no longer a need to carefully craft language to express all of the implementation requirements. There is no longer a need for the test sponsor to have an intimate knowledge of all of the benchmark constraints and their interrelationships. All of this can be captured and expressed cleanly and concisely in the form of code.

In addition to avoiding these complexities, a TPC-provided kit saves development time and costs on the test sponsor's part. This allows a test sponsor to get an environment up and running with less up-front investment. TPCx-HS[17] was the first Express benchmark released by the TPC, with 4 official results published so far.

4.1.1 Software components of TPCx-V benchmark kit

The TPCx-V benchmark specification will be published when the TPC General Council approves the benchmark for official release, and will have a detailed description of the architecture and components of the TPCx-V benchmark kit. We will provide a brief description here.

There are five software components to the TPCx-V benchmark driver; four that are used to drive the workload and one to provide reporting functionality:

- **Prime client**: The prime client (vdriver.jar) is the benchmark execution controller. It coordinates and controls the behavior of the CE client(s), MEE clients(s) and Tier A SUT connectors through RMI connections to each.
- **CE client**: The client emulator (vce.jar) is responsible for emulating customers, requesting a service of the brokerage house, providing the necessary input for the requested service, etc.
- **MEE client**: The market exchange emulator (vmee.jar) is responsible for emulating the stock exchanges by providing services to the brokerage house, performing requested trades, providing market activity updates, etc.

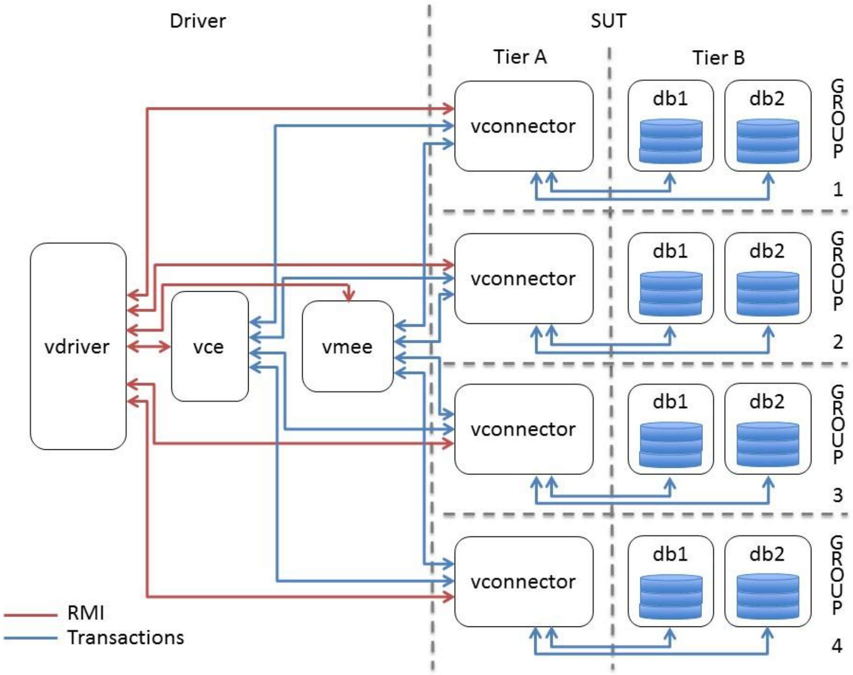

Fig. 3. Software architecture (for a single Tile) of the TPCx-V benchmark kit

- **Tier A SUT connector**: The Tier A SUT connector (vconnector.jar) receives the transaction requests from the CE and MEE clients and sends queries to its Tier B databases.
- **Reporter**: The reporter (reporter.jar) performs the self-validation checks against the transaction log data and (optionally) creates an executive summary report.

Fig. 3 illustrates the four benchmark driver components and the communication paths between them. The RMI communication is used to control and coordinate benchmark runtime behavior while the actual benchmark transactions occur on separate network ports and/or hosts.

4.2 Self-validation

In order to minimize the auditing requirements for this benchmark as well as to help the benchmark user more readily identify run validation errors prior to engaging in a benchmark result audit, the TPCx-V benchmark includes self-validation code in the reporter that checks as many of the validation requirements as possible. These validation checks include:

- **Sampling interval data**: The tpsV sampling data required to create the test run graph referenced in clause 6.8.2 of the benchmark specification
- **Input mix checks**: The input value mix requirements specified in clause 6.5.1 of the benchmark specification
- **Transaction mix checks**: The transaction mix requirements specified in clause 6.4.1 of the benchmark specification
- **Average response time checks**: checks that the average response time for each type of transaction is not greater than the corresponding 90th% response time per clause 6.6.1.4 of the benchmark specification
- **Group measured throughput checks**: checks that the Group measured throughput for each phase is between 98% and 102% of the expected throughput for that group per clause 6.8.1.3 of the benchmark specification

The reporter writes out a list of each of the validation checks performed and whether the benchmark result passed or failed that check.

4.3 Self-audit

One of the longstanding positive characteristics of all TPC benchmarks is the rigor that is applied in validating each implementation. Traditionally this is accomplished via independent third party review performed by TPC-Certified auditors.

The TPC Express model requires the use of a TPC-provided kit. As a result, all implementations now have much more in common. Thus it is feasible to include a set of tools with the kit that automate, or at least facilitate, some of the required audit tasks. Since the TPC-provided kit limits which DBMS may be used, tools have be written to facilitate many audit tasks related to the database.

- The database schema tool captures details of all user-defined types in every database.
- The cardinality tool captures the current cardinality of all TPCx-V tables in each database in the testbed. This data is used to validate the state of an initially populated database, the state of a database prior to any given test run, and it can be used as the basis for space calculations. Cardinality tests are run in parallel, and the outcomes are hierarchically rolled up from the individual database level up to the SUT level
- The ***atomicity*** tool is used to validate that commit and rollback control operations are handled correctly by the DBMS.
- The database population rules and transaction profiles create a set of conditions that should always be true. The specification defines 3 separate conditions that are to be evaluated. The ***consistency*** tool is used to validate that these three conditions are met in all databases in the testbed.
- The TPCx-V specification defines ***isolation*** levels that must be maintained for each of the transactions in the workload. Furthermore, it defines three tests that must be performed to ensure that these required isolation levels are met. The tests purposely create conflicts between concurrently executing transactions and thereby show the ability to handle the conflicts correctly. The isolation tool implements the required tests, captures the necessary data, and reports whether the conditions were met.

5 Experimental results with TPCx-V

To illustrate tuning with TPCx-V, we will use two examples: one that compares a non-varying load with TPCx-V's elastic load, and one that was run on an untuned configurations. First, let us briefly consider the configuration used for testing.

5.1 Testbed configuration

5.1.1 Benchmark

For this set of experiments, we wanted to create as difficult a challenge for the virtualization platform as we could. So although the server would have normally had 1 or 2 Tiles based on its performance, we built a 5-Tile configuration with 60 VMs. We also loaded as many LUs as the disk drives had space for, which gave us a total of 800 LUs, divided into 5 Tiles of 160 LUs each, with each Group 1/2/3/4 having 16/32/48/64 LUs.

5.1.2 Hardware

- HP ProLiant DL580 G7
  - 4 Intel(R) Xeon(R) CPU E7- 4870 @ 2.40GHz processors
  - 40 cores/80 threads
  - 512GB of memory
- Two EMC VNX.5700 disk arrays:
  - Storage Processors with Intel Xeon Dual Core 5600 CPUs and 18GB of memory
  - 72 SSDs for the tables of DSS VM2s, which have high IOPS requirements
  - 112 spinning 15K RPM drives for the tables of OLTP VM3s
  - 10 spinning 15K RPM drives for PostgreSQL redo logs

As we will see in section 5.2, the key to optimizing performance for TPCx-V (and indeed, for a multi-tenant server in the cloud) is to spread the entire load equally across all the resources. When one tenant is hitting a peak, another one might be experiencing a low-load period, allowing the system resources to keep up with the demand. Following this policy, all the data from all the VMs were striped across all the disk drives.

5.1.3 Software

One of the benefits of virtualization is that a virtual computer can be abstracted as a *file*, and be moved or copied. A common use of this property is to package and distribute applications as self-contained virtual appliances. The TPCx-V subcommittee has created a downloadable VM template in the OVF[4] format with all the necessary software for the benchmark pre-loaded and pre-configured. Although the use of this template is not mandatory, using it greatly reduces the benchmark installation time.

The tests were run on VMware vSphere version 6.0, plus the following software stack required by the benchmark specification:

- Red Hat Enterprise Linux 7.1 (3.10.0-123)
- PostgreSQL 9.3
- unixODBC-2.3.1-10
- Java jdk1.7.0_71
- TPCx-V source code version 242 from the TPC subversion server

5.1.4 Virtual Machines

The 60 VMs were cloned from the OVF file described in section 5.1.3. We used PowerCLI[18] scripts to customize each VM to have a different number of virtual CPUs and a different memory size. *vmdk* virtual disks were created in advance, and were added to Tier B database VMs using PowerCLI scripts.

5.2 Results

Valid TPCx-V test runs are 10-Phase, 2-hours runs. But we also ran 2 hours without any Phase changes to have a baseline for investigating the effects of load elasticity.

To aid in locating the plotted graphs (Excel *series*) in the figures, the legend for each figure lists the series in the order that they appear at the leftmost portion of the figure.

5.2.1 Grouping VMs by Group or by Tile

To study the profile of individual components of the System Under test (SUT), throughput values can be calculated and plotted for each VM of each Group of each Tile. In our configuration, that would mean 60 such graphs. However, that is unnecessary. Recall from section 3.3.3 that the benchmark kit guarantees that all 5 Tiles have the same throughput using a deck of cards method. The same is true for the 4 Groups of a Tile: the kit ensures that their offered loads and resulting throughputs conform exactly to the specified ratios. Similarly, the DSS and OLTP VMs will receive the proper ratio of transactions. So, if we have the overall throughput plot, adding the per-Tile throughput plots is not *interesting*: each is receiving exactly 1/5th of the overall load. But grouping the results on a per-Group basis presents interesting results since each Group receives a different proportion of the load, which varies for the group from Phase to Phase.

5.2.2 Single Phase results

For the single-Phase tests, the load contribution of Groups 1/2/3/4 remained constant at 10%/20%/30%/40% as seen in **Fig. 4**. Although this avoids elasticity and is clearly not acceptable for publishing TPCx-V results, we ran this test to create a *baseline* to study the effects of elasticity. For **Fig. 4**, we have added the contributions from Group 1s of all 5 Tiles together. We can see that these 5 Groups together contribute 10% of the overall throughput. Groups 2, 3, and 4 similarly contribute 20%, 30%, and 40%, respectively. **Fig. 5** shows that the sum of CPU utilizations of all 60 VMs, which averages to 5,777%. The server has 40 cores/80 hyperthreads, and reports an average utilization

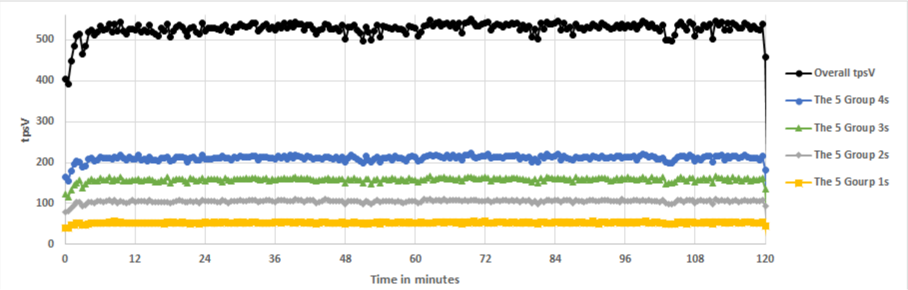

Fig. 4. Run without elasticity: throughputs of 5 Groups 1/2/3/4, summed over the 5 Tiles

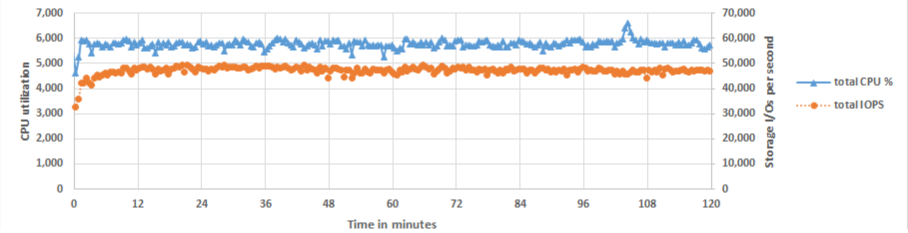

Fig. 5. Total CPU usages and I/O rates for run without elasticity

of 73% for each thread for a total of 5840%. The small difference is due to some processing inside the hypervisor that is not recorded by the VMs. The overall I/O rate is 47K IOPS. When we add the CPU utilizations of all Group 1 VMs together, they amount to 550%, around 1/10th of the total as expected from these Groups contributing 1/10th of the total throughput. **Fig. 7** shows a similar behavior for Groups 2-4.

Fig. 6 shows the CPU utilizations of the 3 VMs of Tile 1, Group 1. Tier A VM1 has the lowest CPU utilization as expected. Tier B VM2 is also low in CPU utilization at around 20% average. Tier B VM3 comes in at around 69%. The situation is reversed for I/O where VM2 has 608 reads/sec and 84 writes/sec, whereas VM3 sees only 97 reads/sec and 140 writes/sec.

5.3 10-Phase results

For these runs, we allowed the kit to vary the load offered to each Group based on the elasticity requirements of the TPCx-V specification. We will study the effects of elasticity on performance by first reporting the result from runs on an early, unoptimized configuration. We will show how TPCx-V identified the source of a performance problem, and will demonstrate the effects of the optimization step.

5.3.1 Results on unoptimized configuration

In this early configuration, we had allocated the CPU counts listed in **Table 1** for the 12 VMs of each Tile. The throughput of this run was 482 tpsV, which is 91% of the single-phase throughput of the run in section 5.2.2. This is a poor result since we want to showcase how well the virtualization platform can handle the cloud-like variations

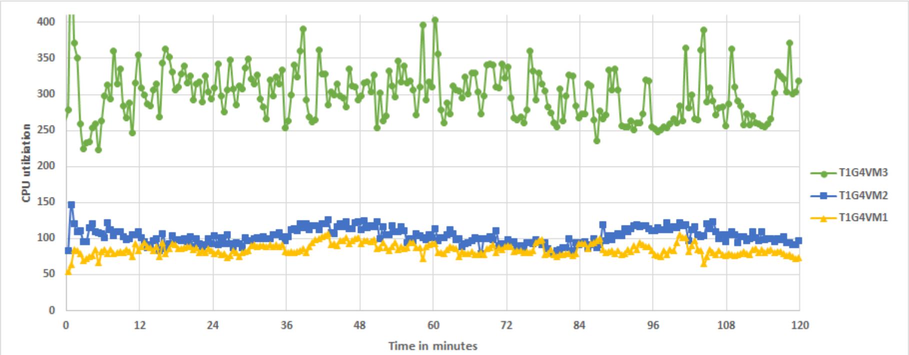

Fig. 6. CPU usage for 3 VMs of Tile 1 Group 1 for run without elasticity

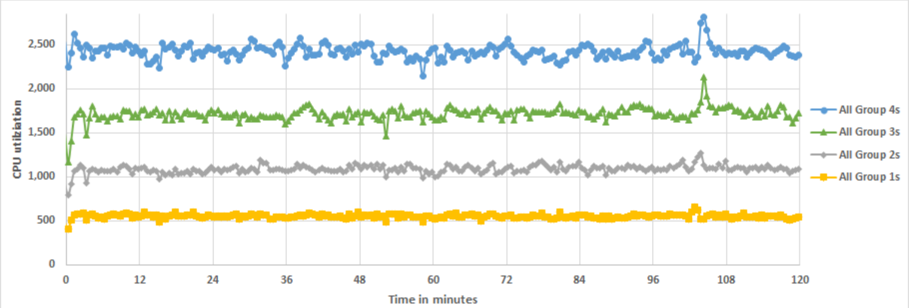

Fig. 7. CPU usage summed by Group for run without elasticity

in load. To see the source of this performance drop, let us first consider the throughput curves in **Fig. 8**. We can see that the overall throughput stays over 500 tpsV until Phase 7, which starts at minute 72. At this point, when Groups numbered 1 in each Tile reach their peak demands, we have a slight drop in performance. The drop is more pronounced in Phase 8 when Groups numbered 2 reach their peak. We see in **Table 1** that the CPU-heavy VM3 of Group 1 has 3 virtual CPUs. The CPU utilization graph of T1G1VM1 in **Fig. 9** (in yellow) shows an average utilization of over 200% and even hitting 300% during Phase 7. If there are any transient peaks, T1G1VM3 may not be able to satisfy the demand. The situation is more pronounced in Phase 8 for T1G2VM3 (in green) with 4 vCPUs and average utilizations that regularly approach 400%.

5.3.2 Optimized results

We increased the virtual CPU count of VM3s in all Group 1s to 4 vCPUs, and VM3s in all Group 2s to 5 vCPUs, and repeated the experiment. **Fig. 11** shows that although

Table 1. virtual CPU counts of VMs

Group	Group 1			Group 2			Group 3			Group 4		
VM	VM1	VM2	VM3	VM1	VM2	VM3	VM1	VM2	VM3	VM1	VM2	VM3
vCPUs	1	1	3	1	2	4	1	2	6	2	2	6

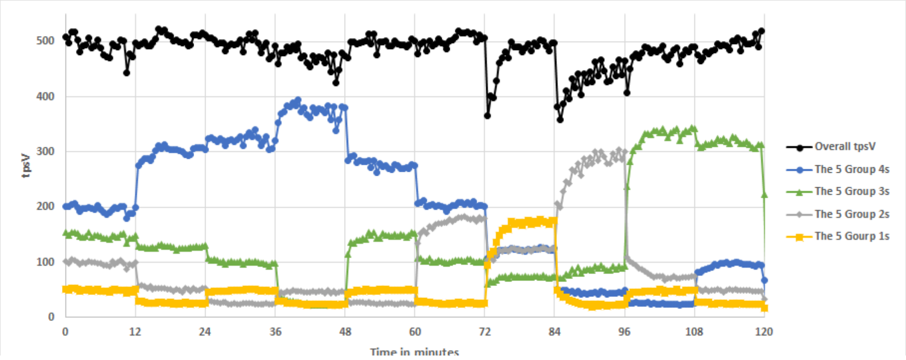

Fig. 8. Overall tpsV and per-Group throughputs

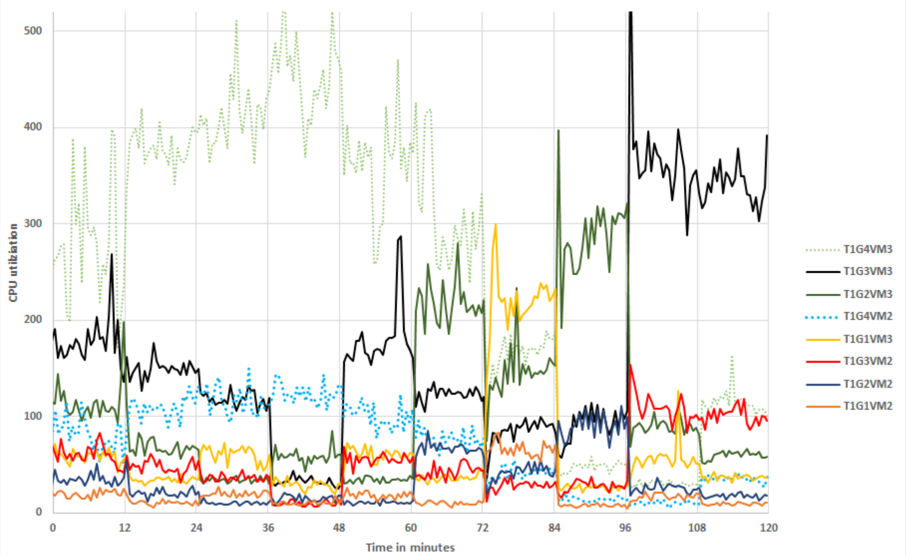

Fig. 9. CPU utilization of the 8 Tier B VMs of Tile 1

there is a drop in throughput in Phases 8-10, it is not nearly as pronounced as in **Fig. 9**. In **Fig. 11**, we can see that T1G2VM3 can use more than 400% of CPU time in Phase 8, so allocating 5 virtual CPUs to it ensured that it will always meet transient demand peaks, as did allocating 4 virtual CPUs to T1G1VM3.

The throughput of this run was 512 tpsV, 6% higher than the earlier run, and within 3% of the run with no elasticity.

6 Performance analysis of results

6.1 Overcommitment and elasticity

Optimizing a configuration for TPCx-V requires CPU *overcommitment*, a feature in wide usage in cloud environments. If we add up the CPU counts of all 60 VMs in the

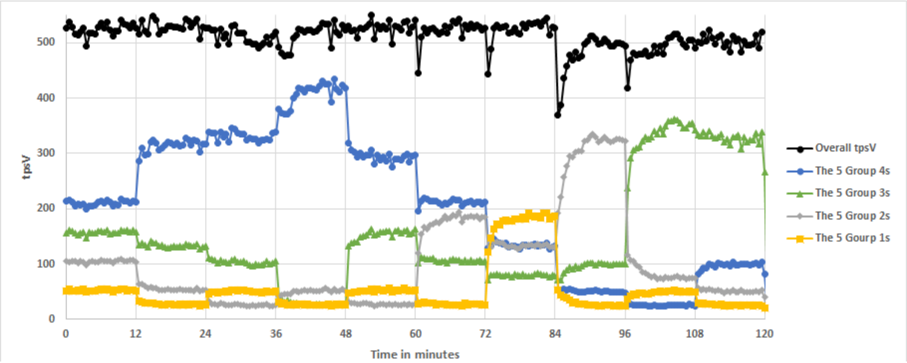

Fig. 10. Overall tpsV and per-Group throughputs after optimization

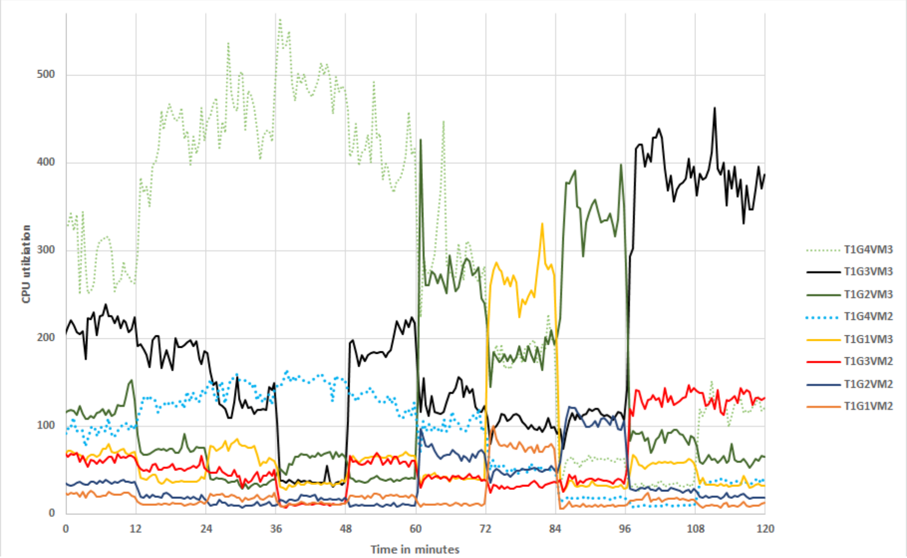

Fig. 11. CPU utilization of the 8 Tier B VMs of Tile 1 after optimization

optimized case, we arrive at a total of 165. The server has only 80 hyperthreads. This is called overcommitment of CPU resources. We cannot expect to have all the VMs running at max utilization at once. However, allocating only enough virtual CPUs to handle the average load will leave the VM under-provisioned during its peak demand periods. So to optimize performance, we need to provision the virtual CPUs of each VM based on peak demand.

Overcommitment of virtual CPUs works well when the peak periods of one VM match up with the low periods of another VM (or the peak demand period of one tenant in a cloud server coincides with a low demand period of another). TPCx-V emulates this characteristic. We can see in **Fig. 10** that the benchmark injects a nearly constant overall load, and as long as the virtualization platform is well-optimized, the server should be able to handle the load despite the wide variations of the load of each VM.

6.2 Hysteresis

The TPC-E workload, which is the origin of the TPCx-V workload, emulates a brokerage house. For one of its main transactions, Trade-Order, around 20% of orders are *limit* orders, and are deferred until the limit price is reached. In TPCx-V, the limit price is guaranteed to be reached within 6 minutes. After running for a while and reaching steady state, an equilibrium exists where the rate of new limit orders that are deferred matches exactly the rate of old limit orders that reach their intended price and are executed. In other words, previously-deferred limit orders make up 20% of executed Trade-Result transactions, while 20% of new Trade-Order transactions are deferred. At any given point in time, the average number of deferred Trade-Order transactions is 24Xtps. So, for example, running at 186 tps, an average of 37.2 transactions per second are deferred; and there are an average of 4,464 transactions in the deferred queue, waiting for their limit price to be reached.

In **Fig. 12** we see Tile 1, Group 1transitioning from Phase 6, when it is running at around 26.6 tps, to Phase 7, where it will eventually run at 186 tps. Its contribution grows from 5% to 35%. But the number of deferred limit orders at the beginning of Phase 7 is only 24X26.5=636. As these orders meet their limits and are completed, their contribution to the throughput is not at the same rate that 4,464 transactions in the deferred queue (corresponding to a 186 tps throughput) would have provided. Hence, we start at 122 tps, and it takes 6 minutes for all the limit orders from Phase 6 to be drained, before the Group runs at its steady state 186 tps for the next 6 minutes. The situation is reversed in the transition to Phase 8 where the Group wants to run at 26 tps again, but has a large backlog of 4,464 limit orders from Phase 7. It takes 6 minutes before throughput drops to the desired level.

Although each Group has to deal with this hysteresis effect at every Phase transition, the too-high and too-low hysteresis effects of the various Groups should match up and cancel out, and the overall throughput should not be impacted if the SUT is well-optimized and the virtualization platform is efficient.

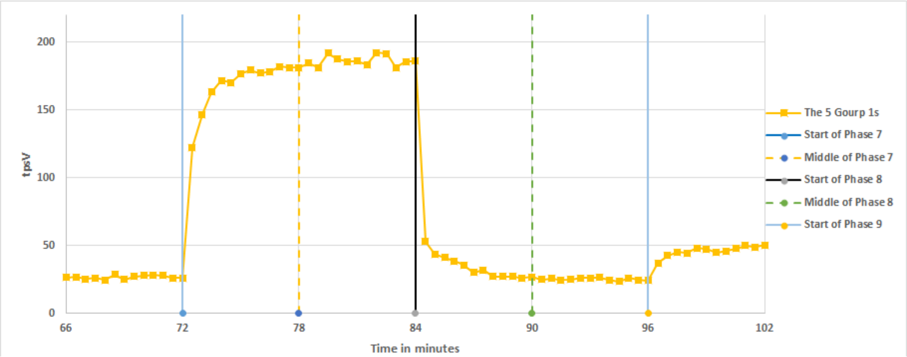

Fig. 12. Overall tpsV and per-Group throughputs after optimization

7 Future work

Potential areas of future work with the TPCx-V benchmark involve expanding the database coverage of the benchmark and leveraging the workload and its characteristics to measure the performance of cloud and cloud infrastructure environments.

7.1 Database coverage

Adding support to the TPCx-V kit for databases other than PostgreSQL would make the kit attractive to a wider audience. The use of the ODBC client API in the kit was a design choice to make it easier to add other databases. A logical next step would be to add support for the MySQL interface to the TPCx-V kit. Having this interface in the kit would immediately add support for many database environments such as MySQL, MariaDB, Percona, and others that support the MySQL interface. Having the ability to drive a TPCx-V load against these additional database environments would make the TPCx-V kit and benchmark interesting to individuals, academic institutions, and companies that are more familiar with these other database environments.

7.2 Cloud and Cloud infrastructure

The TPCx-V workload and kit were designed to drive and measure the performance of multiple distinct database environments. The TPCx-V benchmark specification states that all these environments must be run on a single virtualized server. However, the usage of the TPCx-V kit could be expanded to measure the performance of environments where the Tiles/Groups/VMs can be placed on multiple servers, and the elasticity features of the benchmark can be used to measure the efficiency of the testbed in deploying and possibly migrating VMs and applications as the load changes and need arises. In other words, a better emulation of cloud data centers and cloud infrastructure. Also, since the TPCx-V kit does not inherently need to know the location or placement of the databases it is driving, the kit could be used to drive elastic database workloads in public cloud environments where the details of the underlying implementation are typically abstracted.

8 Acknowledgements

We thank Matt Emmerton, John Fowler, Karl Huppler, Matthew Lanken, Jamie Reding, Cecil Reames, Jignesh Shah, Wayne Smith, and Priya Sethuraman for contributing time and effort in the development of the TPCx-V benchmark. We are also grateful to the reviewers for their comments and corrections.

9 References

1. Andrew Bond, Greg Kopczynski, and H. Reza Taheri: 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: Raghunath Nambiar, Meikel Poess (Eds.): Selected Topics in Performance Evaluation and Benchmarking - 4th TPC Technology Conference, TPCTC 2012, Springer 2013, LNCS Volume 7755, ISBN 978-3-642-36726-7: pp. 34-50
2. Andrew Bond, Doug Johnson, Greg Kopczynski, and H. Reza Taheri: Architecture and Performance characteristics of a PostgreSQL implementation of the TPC-E and TPC-V workloads. In: Raghunath Nambiar, Meikel Poess (Eds.): Selected Topics in Performance Evaluation and Benchmarking - 5th TPC Technology Conference, TPCTC 2013, Springer 2013, LNCS Volume 8391, ISBN 978-3-319-04935-9.
3. Eric Deehr, Wen-Qi Fang, H. Reza Taheri, and Hai-Fang Yun: Performance analysis of database virtualization with the TPC-VMS benchmark. . In: Raghunath Nambiar, Meikel Poess (Eds.): Performance Characterization and Benchmarking. Traditional to Big Data - 6th TPC Technology Conference, TPCTC 2014, Hangzhou, China, September 1-5, 2014. Revised Selected Papers. Lecture Notes in Computer Science 8904, Springer 2015, ISBN 978-3-319-15349-0
4. Distributed Management Task Force: Open Virtualization Format Specification, Version 1.1.0, 2010-01-12.
5. Renato Figueiredo, Peter A. Dinda, and Jose A. B. Fortes "Guest Editors' Introduction: Resource Virtualization Renaissance," *Computer*, vol. 38, no. 5, pp. 28-31, http://www2.computer.org/portal/web/csdl/doi/10.1109/MC.2005.159, May 2005.
6. Susanta Nanda and Tzi-cker Chiueh, “A Survey on Virtualization Technologies.” *Technical Report ECSL-TR-179*, SUNY at Stony Brook, Feb. 2005.http://www.ecsl.cs.sunysb.edu/tr/TR179.pdf
7. Mendel Rosenblum and Tal Garfinkel: "Virtual Machine Monitors: Current Technology and Future Trends," *Computer*, vol. 38, no. 5, pp. 39-47, May 2005
8. Priya Sethuraman and H. Reza Taheri: TPC-V: A Benchmark for Evaluating the Performance of Database Applications in Virtual Environments: In: Raghunath Nambiar, Meikel Poess (Eds.): Performance Evaluation and Benchmarking: TPCTC 2011: LNCS vol. 6417, Springer 2011, ISBN 978-3-642-18205-1 pp. 121-136
9. Greg Smith, *PostgreSQL 9.0 High Performance*, Packt Publishing (October 20, 2010).
10. SPECvirt_sc2010 benchmark info, SPEC Virtualization Committee: http://www.spec.org/virt_sc2010/
11. SPECvirt_sc2013 benchmark info, SPEC Virtualization Committee: http://www.spec.org/virt_sc2013/
12. VMware, Inc. http://www.vmware.com/products/vmmark/overview.html
13. IDC: “Worldwide Virtual Machine 2013–2017 Forecast: Virtualization Buildout Continues Strong”, http://www.idc.com/getdoc.jsp?containerId=242762.
14. SPECvirt_sc2013 benchmark info, SPEC Virtualization Committee: http://www.spec.org/virt_sc2013/
15. TPC: Detailed TPC-E Description: http://www.tpc.org/tpce/spec/TPCEDetailed.doc
16. TPC: TPC-VMS benchmark: http://www.tpc.org/tpcvms/default.asp
17. TPC: TPCx-HS benchmark: http://www.tpc.org/tpcx-hs/default.asp
18. VMware, Inc., PowerCLI documentation, https://www.vmware.com/support/developer/PowerCLI
19. VMware, Inc., VMmark 2.x, http://www.vmware.com/products/vmmark/overview.html