Cloud Performance Benchmark Series

Amazon Relational Database Service (RDS)
TPC-C Benchmark

Md. Borhan Uddin
Bo He
Radu Sion

ver. 0.6

1. Overview

Experiments were performed to benchmark the Amazon Relational Database Service (RDS) within a TPC-C benchmarking framework. The TPC-C benchmark is one of the most widely adopted database performance benchmarking frameworks comparing OLTP performance of online transaction processing systems. Two types of Amazon RDS services were tested, namely the standard RDS (single availability zone) and the Multi-AZ RDS (synchronous 'standby' replica in multiple availability zones). For each service type, five different RDS instances were tested: Small, Large, Extra Large (XLarge), Double Extra Large (2XLarge), and Quadruple Extra Large (4XLarge).

2. Setup

The Amazon Relational Database Service (RDS) runs full-featured MySQL database servers (version 5.1.45). The deployed benchmark was the open-source tpcc-mysql TPC-C MySQL benchmark instance, developed by the popular MySQL development community Percona.

All of the tested RDS service instances were located on the same Amazon Web Service (AWS) US East-1a region. Extra Large Amazon Elastic Compute Cloud (EC2) machines were used to probe the RDS instances from the same AWS region. EC2-hosted machines were chosen as TPC-C clients in order to minimize the impact of network/bandwidth and other external interference elements associated with extra-cloud clients. Such elements are not the subject of *this benchmark which aims to purely assess the performance of (solely) the RDS service* in isolation.

Custom scripting was developed and deployed on the Extra Large EC2 machines to automate the benchmark and deploy tpcc-mysql to probe the RDS instances, process the benchmarking outputs, and generate summarized analysis data.

The benchmark was initiated by uploading the TPC-C database schema (tables and constraints) to the RDS instances. The database schema is shown in Figure 1. The tables were then populated with data, and the RDS service's TPC-C performance was then measured. The deliverables of the benchmark were the TPC-C throughput in transactions per minute (tpmC).

Amazon offers two types of RDS services. Standard RDS doesn't deploy database replication. Multi-AZ replicates to multiple availability zones synchronously, although the replicated copies are 'standby' only and read operation can't be performed on them. The price of Multi-AZ RDS is double than that of standard RDS. Instance configurations of Multi-AZ RDS are similar to standard RDS.

Notwithstanding the RDS service type, the RDS service can be run on different underlying machine configurations. At the time of testing there were five such configurations: Large, Extra Large (XLarge), Double Extra Large (2XLarge), and Quadruple Extra Large (4XLarge). The configurations vary according to their allocated memory

(RAM) in Gigabytes (GB), I/O capacity, and allocated EC2 Compute Units [ECU; 1 ECU aims to approximate an Intel 1.0-1.2 GHz 2007 Opteron or 2007 Xeon CPU] etc.

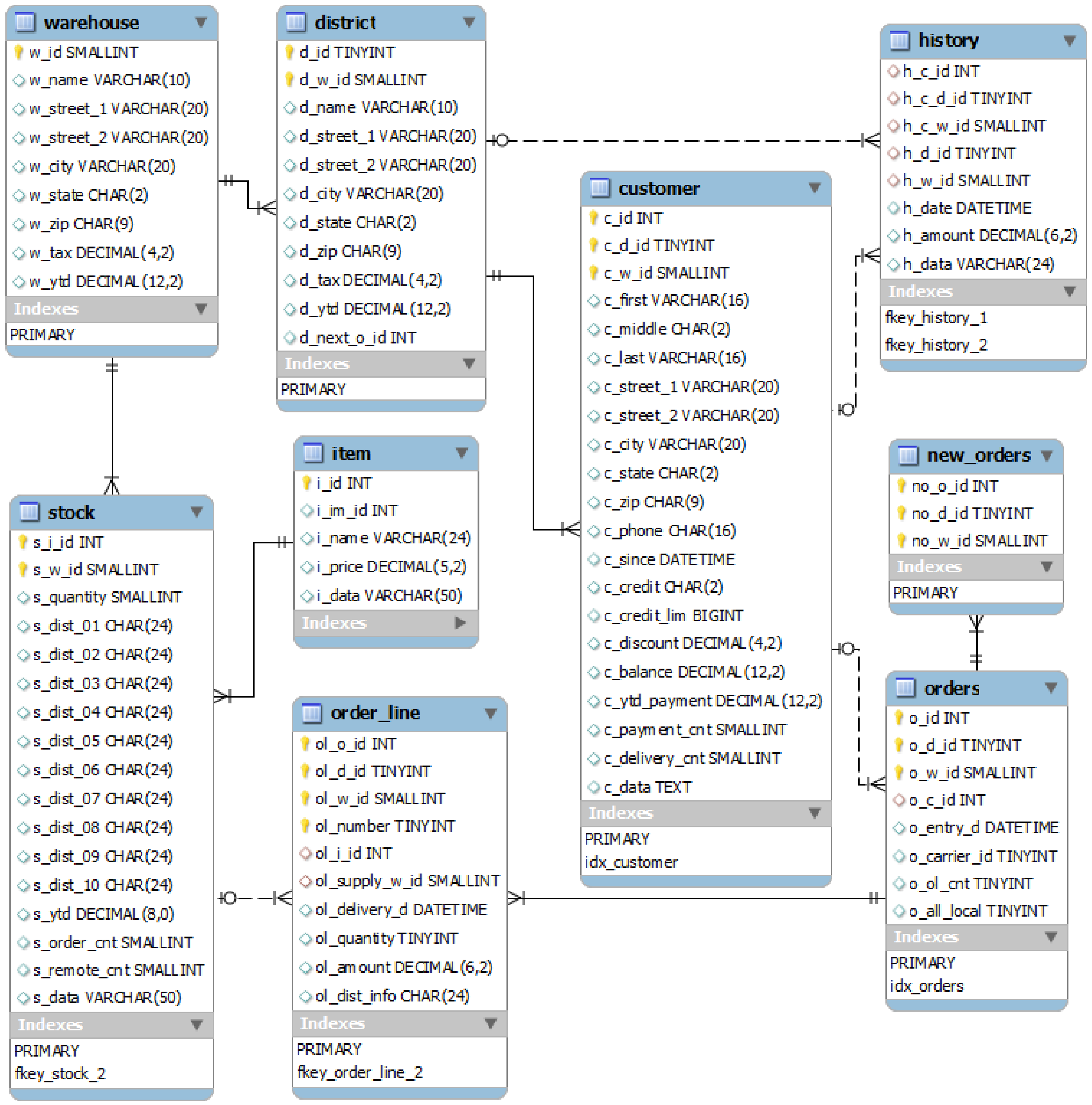


Figure 1: Database Schema of the TPC-C Benchmark (generated by MySQL Workbench 5.2 CE)

The processing power is structured as a set of multiple virtual cores (VC). A VC in turn is considered as a collection of ECUs (and has an associated number of ECUs/VC). Thus, the total number of ECU compute units can be computed by knowing the number of virtual cores and the number of ECUs per virtual core. The parameters corresponding to the testing setup are detailed in Table 1.

Instance Size	RAM (GB)	Total ECU (=VC * ECU/VC)	Platform	I/O Capacity	Region
Small	1.7	1 (=1*1)	MySQL 64-bit	Moderate	US-East-1a
Large	7.5	4 (=2*2)	MySQL 64-bit	High	US-East-1a
Extra Large (XLarge)	15	8 (=4*2)	MySQL 64-bit	High	US-East-1a
Double Extra Large (2XLarge)	34	13 (=4*3.5)	MySQL 64-bit	High	US-East-1a
Quadruple Extra Large (4XLarge)	68	26 (=8*3.25)	MySQL 64-bit	High	US-East-1a

Table 1: Configurations of different RDS Instances

The configuration details of the EC2-hosted probing machines are illustrated in Table 2.

Instance Size	Memory (GB)	Total ECU (=VC * ECU/VC)	Platform	AMI Id	Region
Extra Large	15	8 (=4*2)	Fedora 64-bit	ami-86db39ef	US-East-1a

Table 2: Configurations of the probing EC2 Instances

2.1. Throughput

The incident loads on the RDS serving instances were varied from very low to very high by considering an increasing number of warehouses and concurrent users. Overall, we observed that at a very low load, the resulting throughput was also relatively low; at medium load, the throughput increased to a peak; at very high loads, the throughput decreased again. Over a large number of runs, the targeted instance type was associated with its observed maximum (tpmC).

For each RDS instance, the number of warehouses and users was varied from very low to very high (1 to 2^WH_TH for warehouses, 1 to 2^USR_TH for users respectively). Each experiment segment was run in 2 minute intervals. WH_TH ranged from 0 to 5 and USR_TH from 0 to 10, thus for a typical RDS instance, the number of warehouses varied from 1 to 32 and number of users varied from 1 up to 1024 for select instances.

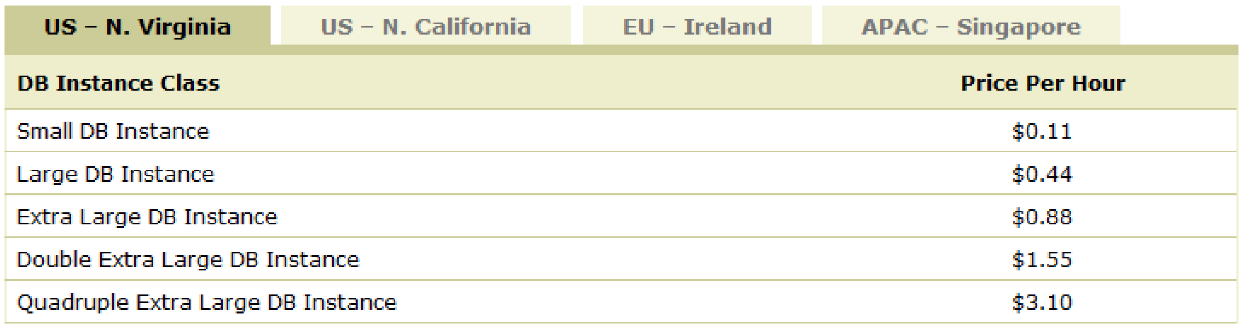
For each measured instance, once the maximum throughput point in the (number of warehouses, number of users) space was observed, its parameters were deployed in a set of longer-lived 30 minute interval measurements to obtain instantaneous throughputs in tpmC. These were then recorded and deployed to compare the RDS instances at their corresponding optimal loads.

2.2. Transaction Cost

In addition to the core tpmC throughput for each RDS service type and instances, the dollar cost per tpmC ($/tpmC - i.e., amount of USD to be paid for 3-year period per tpmC) was also computed for the current Amazon RDS pricing model (included here for reference).

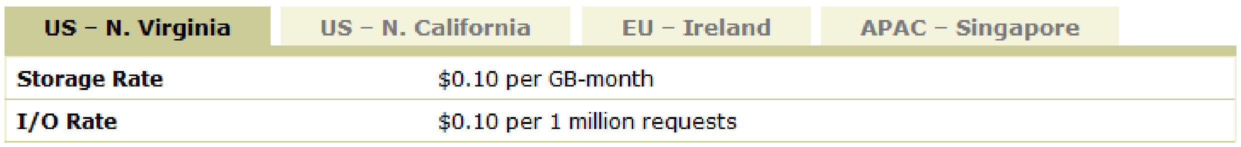
Moreover, it is important to consider the 2 types of RDS instance service levels with different pricing: standard RDS and multi-AZ RDS. For running RDS on Amazon, the

involved costs of four service components needed to be considered: ECU instance cost, storage cost, local I/O cost, and in/egress data transfer cost.
The runtime data storage and I/O transfer costs were collected from the RDS Monitoring service. The data transfer costs were collected using the Amazon RDS Usage Report. The pricing of RDS instances, data storage, I/O & data transfer cost were collected from Amazon for standard and multi-AZ Amazon RDS instance services. Figures 2 and 3 show examples of hourly CPU, as well as storage and I/O pricing for the Amazon On-Demand standard RDS instances.


US – N. Virginia | US – N. California | EU – Ireland | APAC – Singapore

DB Instance Class	Price Per Hour
Small DB Instance	$0.11
Large DB Instance	$0.44
Extra Large DB Instance	$0.88
Double Extra Large DB Instance	$1.55
Quadruple Extra Large DB Instance	$3.10

Figure 2: Pricing (hourly) of Amazon On-demand standard RDS instances (amazon.com)


US – N. Virginia | US – N. California | EU – Ireland | APAC – Singapore

Storage Rate	$0.10 per GB-month
I/O Rate	$0.10 per 1 million requests

Figure 3: Pricing of storage and I/O of Amazon On-demand standard RDS (amazon.com)

We computed the **service provider-related cost of one TPC-C transaction** in USD micro-cents, by considering its individual components, as follows:

transaction cost = (Instance Cost in USD per minute + Storage Cost in USD per minute + Cost of I/O in USD per minute + Cost of data transfer in USD per minute)*10^8/tpmC

The **standard 3-year TPC-C costs** in USD per tpmC (total cost of system running for 3 years divided by tpmC throughput) are defined and computed as follows:

$/tpmC = (Instance Cost in USD per minute + Storage Cost in USD per minute + Cost of I/O in USD per minute + Cost of data transfer in USD per minute)* number of minutes in 3-year (60*24*365*3)/tpmC

3. Results

3.1. Throughput

Total Throughput per Concurrency Level. Figure 4a depicts total throughput as a function of the number of concurrent users, for all of the different number of warehouses in a standard large RDS instance. Low throughput was observed for very low (likely due to under-utilization) and very high (likely due to bottlenecks, saturation and per-user overheads) numbers of concurrent users. Overall Throughput seemed to increase between 16 and 64 concurrent users, peaking at around 64 for most warehouse configurations. Beyond this 64-user point, throughput increases with increasing number of warehouses, yet at a seemingly independent rate.

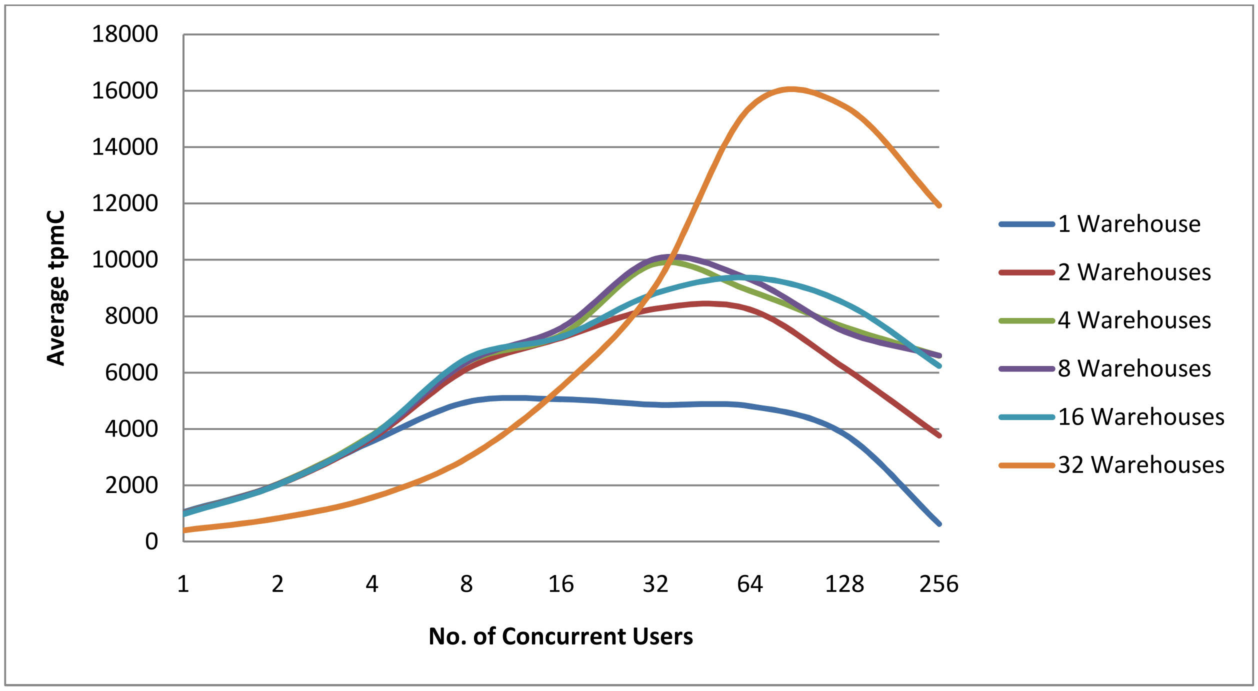


Figure 4a: Throughput (tpmC) of a Standard Large RDS instance setting

Per-user Throughput per Concurrency Level. Figure 4b depicts the average throughput per user as a function of the number of concurrent users. It can be seen that RDS-side context-switching and resource-contention related bottlenecks start to kick in more visibly once 4-8 users are connected. Up to 32 users, for most warehouse configurations perceived throughputs of over 200 tpmC can still be achieved. This suggests the need for a careful evaluation of the specific application needs – if individual processing threads do not require more than a few tens of transactions per minute, it seems that even a single RDS instance can serve up to 64-128 of such clients and there is no reason to purchase additional instances.

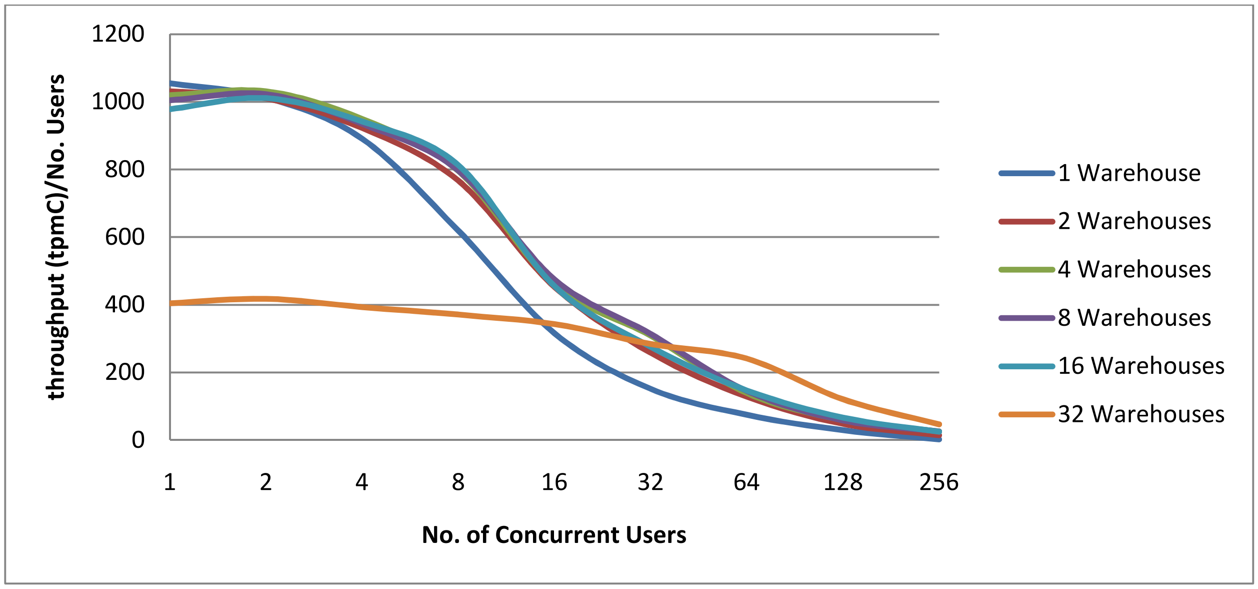


Figure 4b: Average throughput seen by a user with increasing number of concurrent users in a Standard Large RDS instance setting

Throughput per Number of Warehouses. Figure 5 illustrates the behavior of the overall throughput with increasing number of warehouses. For most number-of-users configurations maximum throughput seems to be achievable at a number of warehouses between 4 and 8. Beyond 16 warehouses, the throughputs can be maintained only with a larger number of users, likely due to inter-warehouse context switch and ingress transaction threading overheads – more clients would naturally benefit from the already started warehouse processors. A relatively global optimum seems to have been achieved around 8 warehouses in the considered setup.

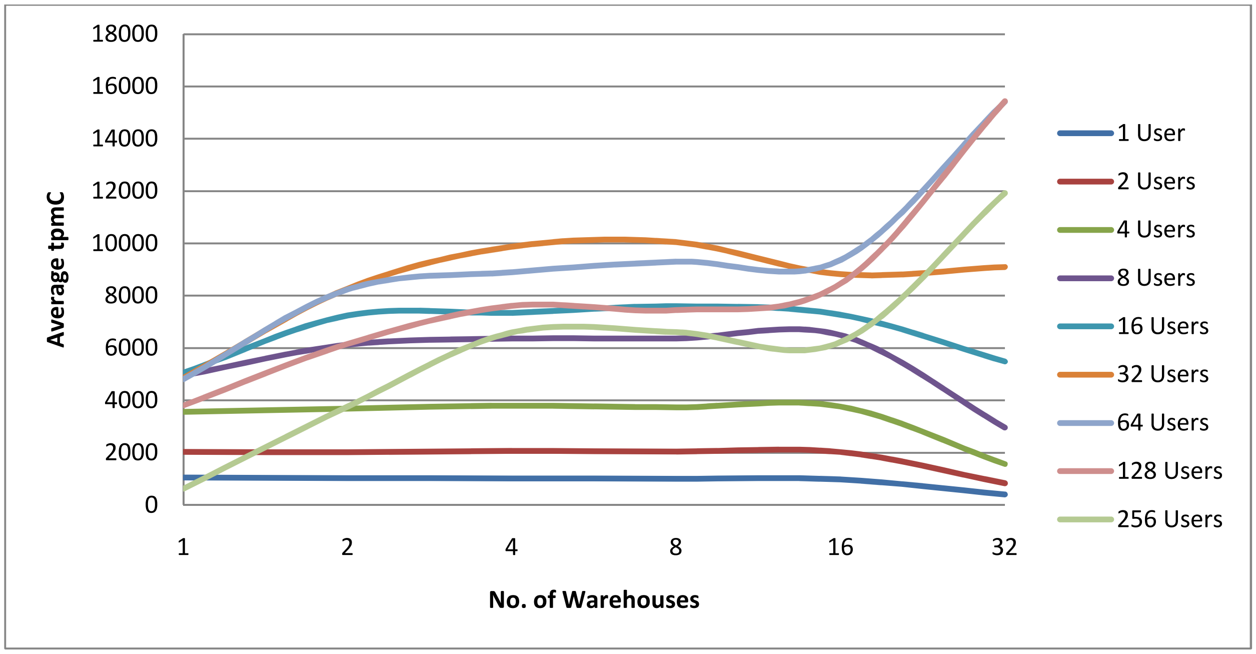


Figure 5: Average tpmC of a Standard Large RDS instance with increasing No. of Warehouses

Optimal [No. of Warehouses, No. of Concurrent Users] Pairs. The pairs featuring maximum throughput we named the 'Optimal' loads for their corresponding RDS target instances. Table 3 shows the optimal loads for all considered RDS instances.

RDS Instance Size	(No. of Warehouses, No. of Concurrent Users)
Small	(8, 32)
Large	(8, 32)
XLarge	(16, 64)
2XLarge	(16, 256)
4XLarge	(8, 64)

Table 3: Optimal load points for standard RDS instances

Detailed Experimental evaluation of throughput at Optimal Load Points. For thoroughness, each RDS instance was also tested for extended period of times at the optimal load points. The average resulting throughputs are shown in Figure 6.

Naturally at optimal load, average throughput is higher for larger instances. The same cannot be said however of the behavior found in large target instance settings. There, interestingly, ***average throughput does not increase significantly with instance size*** – average throughput of XLarge, 2XLarge and 4XLarge are quite similar. This suggests underlying bottlenecks, or possibly the lack of ability of our benchmark to truly saturate the memory (which is the main differentiating factor at this level).

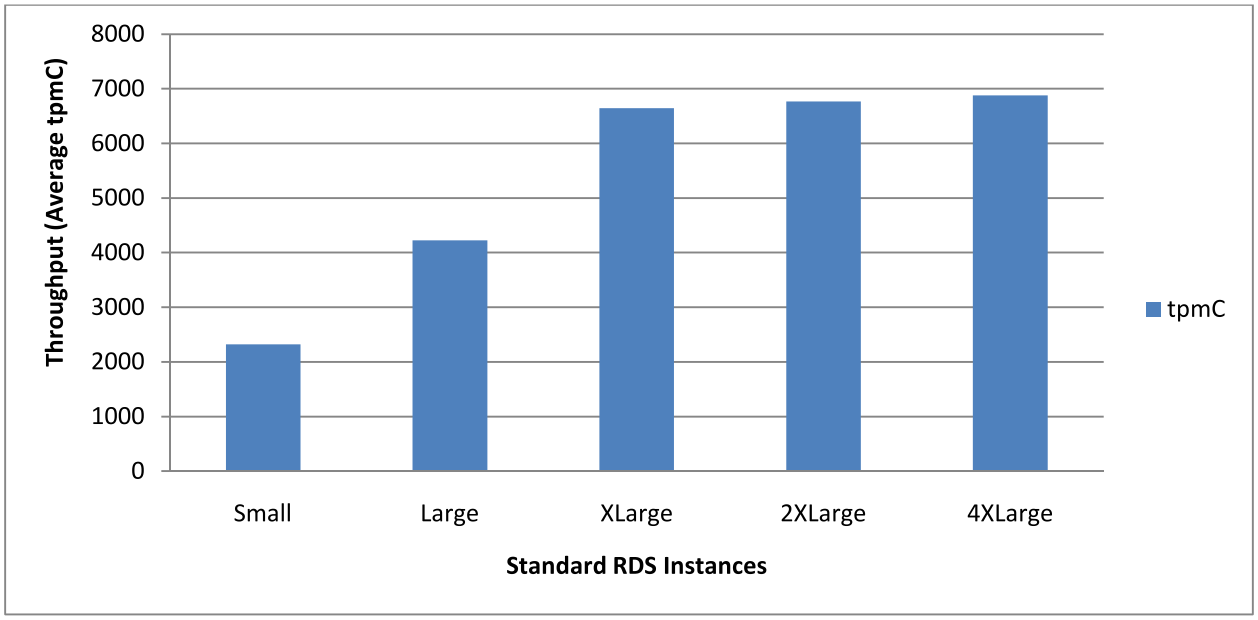


Figure 6: Average throughput (tpmC) of standard RDS instances at optimal load setup

While above aggregates (such as averages) have been illustrated for consistency and control, we considered it interesting to also look at instantaneous at-runtime behavior. Figure 7 illustrates instantaneous throughputs at optimal load points, over a total period of 30 minutes. Similar to the average throughputs seen in Figure 6, the differences between XLarge variants are somewhat negligible. XLarge instances often show the highest instantaneous throughput (around 10K-11K tpmC > 2XLarge and 4XLarge instances), yet feature also the highest jitter factors. This leads to the almost identical average values seen in Figure 6. Overall, throughput increases from Small to XLarge instances, yet beyond XLarge only the jitter factors decrease. Beyond 2XLarge no improvement was seen – neither on throughput nor jitter.

This suggests ***a careful application-specific analysis is required to justify the purchase of any of the XLarge variety instance types***.

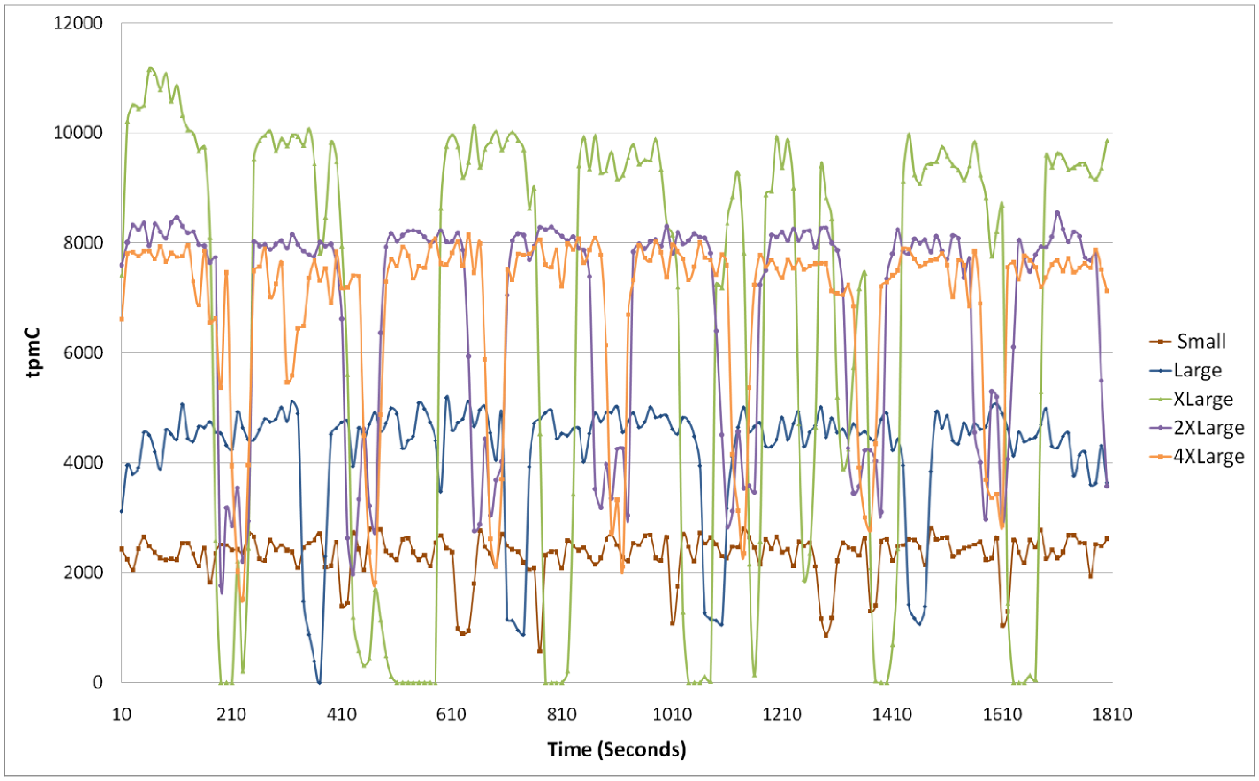


Figure 7: Instantaneous throughput (tpmC) at the optimal load points

Throughput at Fixed Load. To further detail differences between RDS instances, a load point was fixed at (8 warehouses, 64 concurrent users) and average and instant throughputs were measured. Results are shown in Figures 8 and 9.
As previously seen, XLarge, 2XLarge and 4XLarge instances have similar average and instantaneous throughputs.

Overall, *larger instances seem to offer higher throughput with less jitter than smaller instances at similar loads,* ***yet average throughput doesn't increase significantly for larger instances.***

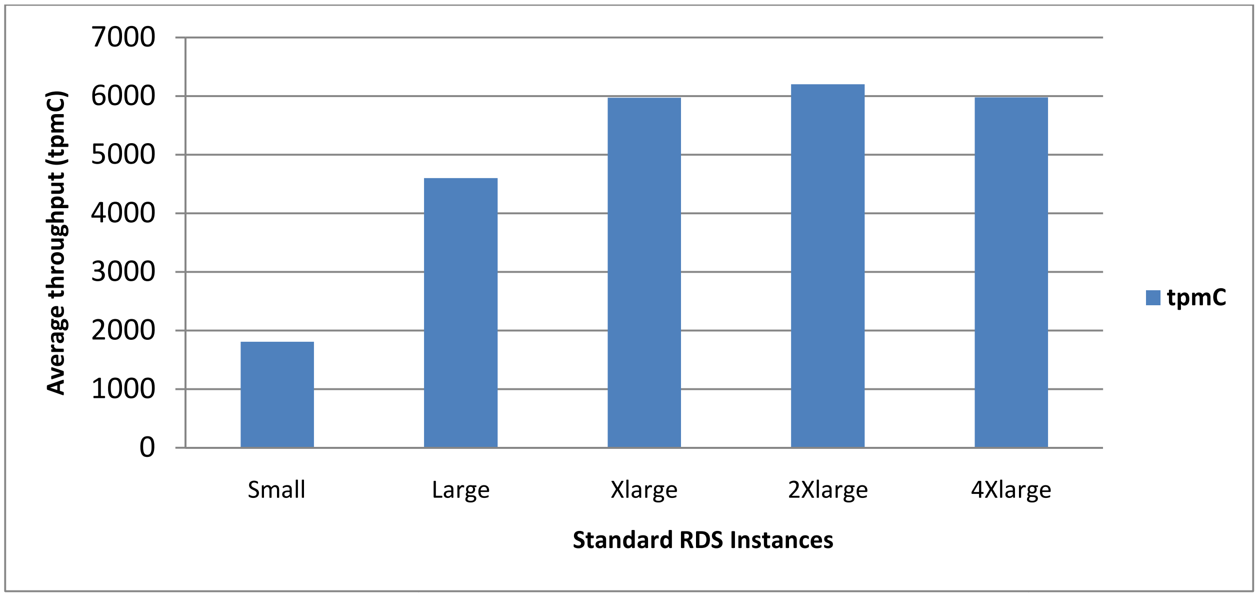


Figure 8: Average throughput (tpmC) at Fixed Load [8 warehouses, 64 users]

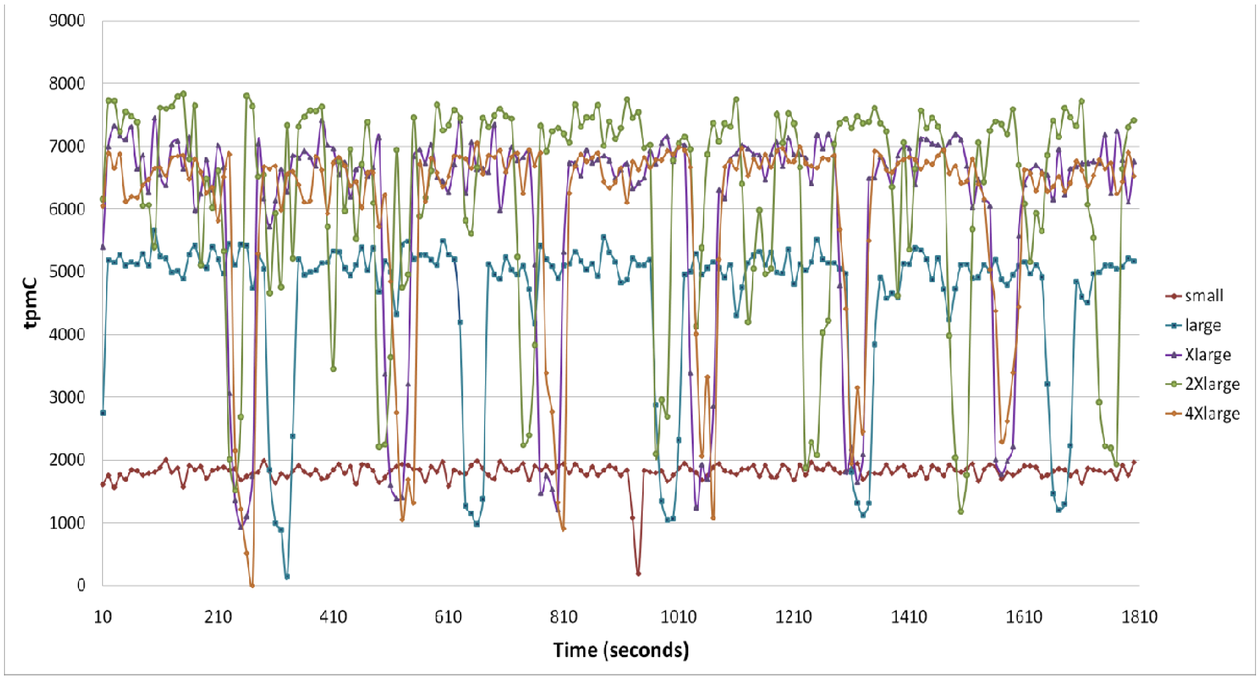


Figure 9: Instantaneous throughput (tpmC) at Fixed Load [8 warehouses, 64 users]

Impact of Replication. To understand the impact of replication, the difference in throughput between standard and multi-AZ RDS instances in the fixed load setup (8 warehouses, 64 concurrent users). Figure 10 depicts the results.

As expected, multi-AZ instances feature ***lower throughputs for all RDS instance types*** (Small – 4XLarge). Moreover the throughput doesn't noticeably increase for larger

instances. We will not speculate why that may be the case beyond the straightforward explanation of a probable bottleneck constituted by the replication mechanism itself.
As a result, the difference in throughput between standard and multi-AZ instances increases for larger instance types. For example standard XLarge-4XLarge throughputs are three times higher than their corresponding multi-AZ variants. For small instances the differences are small.

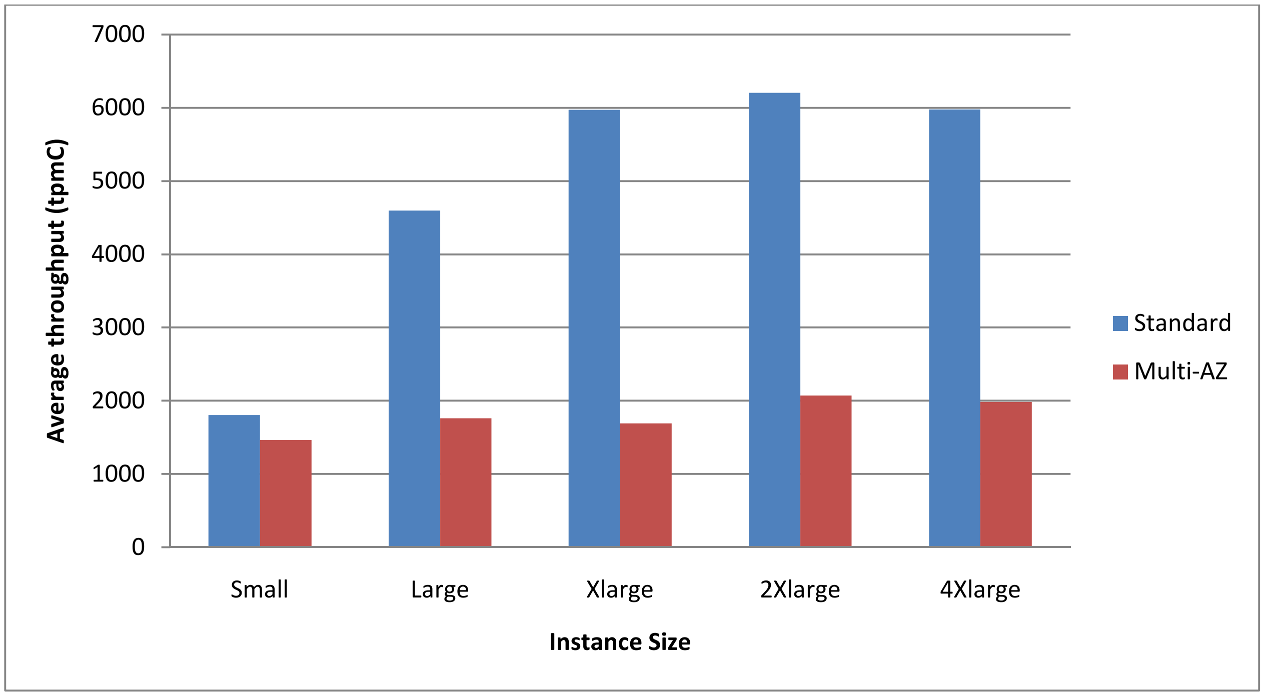


Figure 10: Throughput at Fixed Load (8 Warehouses, 64 concurrent users)

We detailed some of this by looking at instantaneous behavior, depicted in Figure 11 for two different multi-AZ RDS instances: small and 2XLarge. The observed throughputs and jitter factors don't differ significantly.
This leaves open the question of what scenarios justify the purchase of XLarge+ instances. A TPM-C throughput viewpoint does not seem to provide an answer.

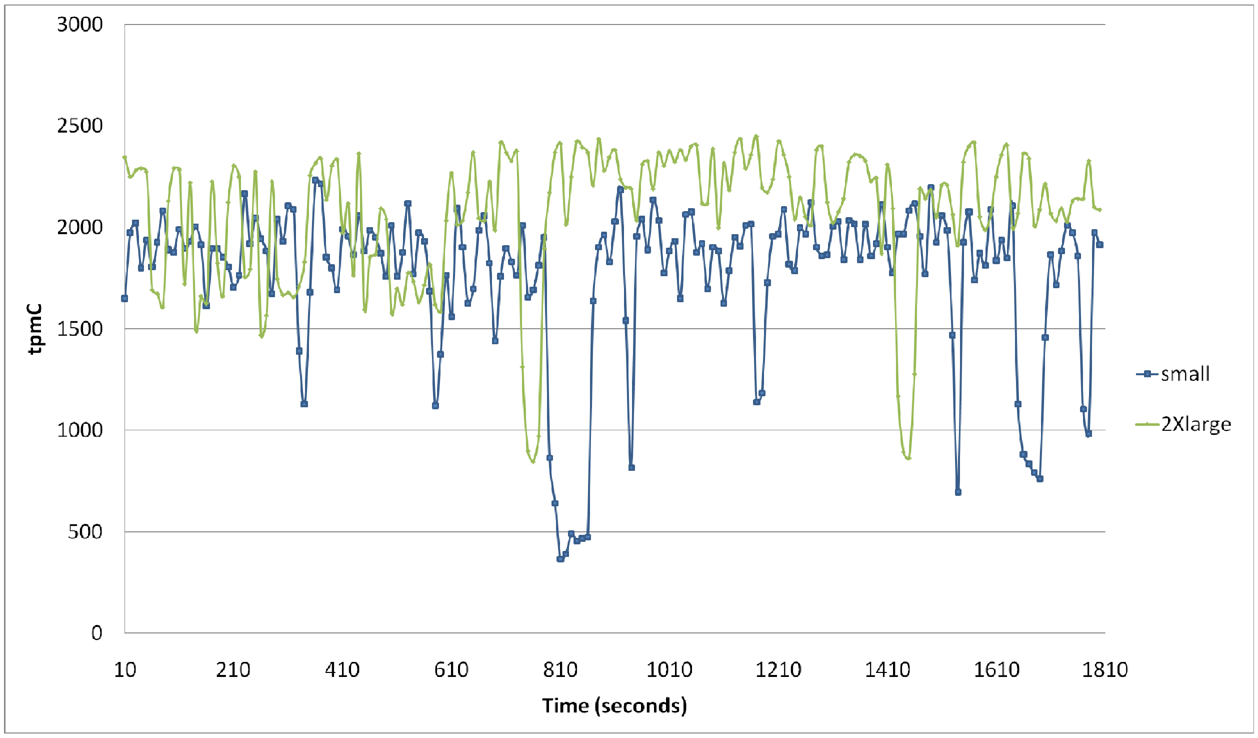


Figure 11: Instantaneous throughput (tpmC) for two different multi-AZ RDS instances at Fixed Load [8 warehouses, 64 users]

3.2. Cost

Table 4 illustrates the calculated costs using the equations discussed in section 2.2. These costs are then illustrated below.

	Standard				Multi-AZ			
Instance	**Instance Price ($/Hour)**	**tpmC**	**µCent/ transactionC**	**$/tpmC**	**Instance Price ($/Hour)**	**tpmC**	**µCent/ transactionC**	**$/tpmC**
Small	0.11	1807.00	206.67	3.26	0.22	1463.80	387.96	6.12
Large	0.44	4597.10	273.10	4.31	0.88	1761.13	996.47	15.71
Xlarge	0.88	5971.40	356.16	5.62	1.66	1693.67	1762.09	27.78
2Xlarge	1.55	6202.80	530.00	8.36	3.10	2070.13	2805.69	44.24
4Xlarge	3.10	5977.60	972.54	15.34	6.20	1985.81	5313.99	83.79

Table 4: Calculated Cost for Standard and Multi-AZ RDS Instances

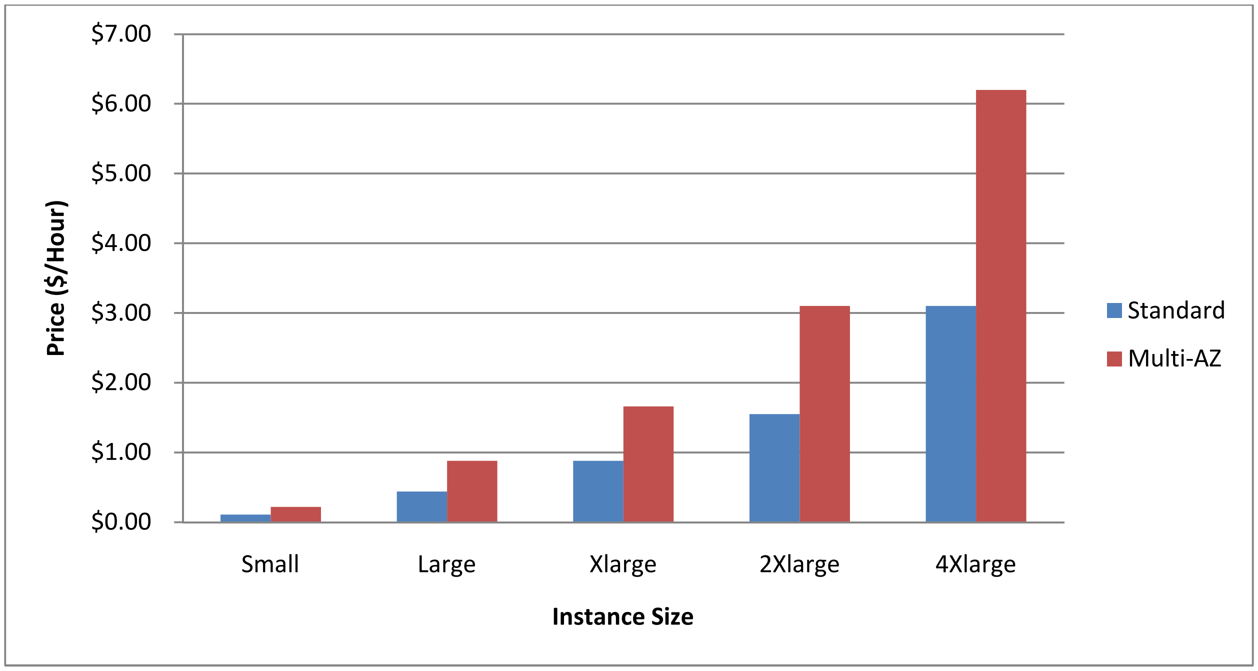


Figure 12: Hourly price of Standard and Multi-AZ RDS Instances (amazon.com)

Figure 12 depicts the Amazon RDS prices for both standard and Multi-AZ service. Standard **3-year TPC-C costs** in USD per tpmC are shown in Figure 13. For both standard and Multi-AZ services, ***$/tpmC increases for larger instances*** which is not necessarily desirable or intuitive. It suggests either a ***pricing inefficiency*** or that the main cost-drivers are not the RDS compute node resources but rather some other hidden factors related to the size of the large instances.

Moreover, while for smaller types (Small to XLarge) the rate increases slowly, for the XLarge to 4XLarge instance types it skyrockets – mainly because of the stagnant observed throughputs beyond XLarge instances (Figure 10) despite the increasing instance costs (Figure 12). In terms of $/tpmC, it seems that standard Small, Large and XLarge instances are more economic than 2XLarge and 4XLarge.

Naturally Multi-AZ service features higher $/tpmC cost. The cost differences between standard and multi-AZ instance types increase with instance size. Thus overall, ***larger instances seem less economic*** for Multi-AZ RDS service in terms of $/tmpC.

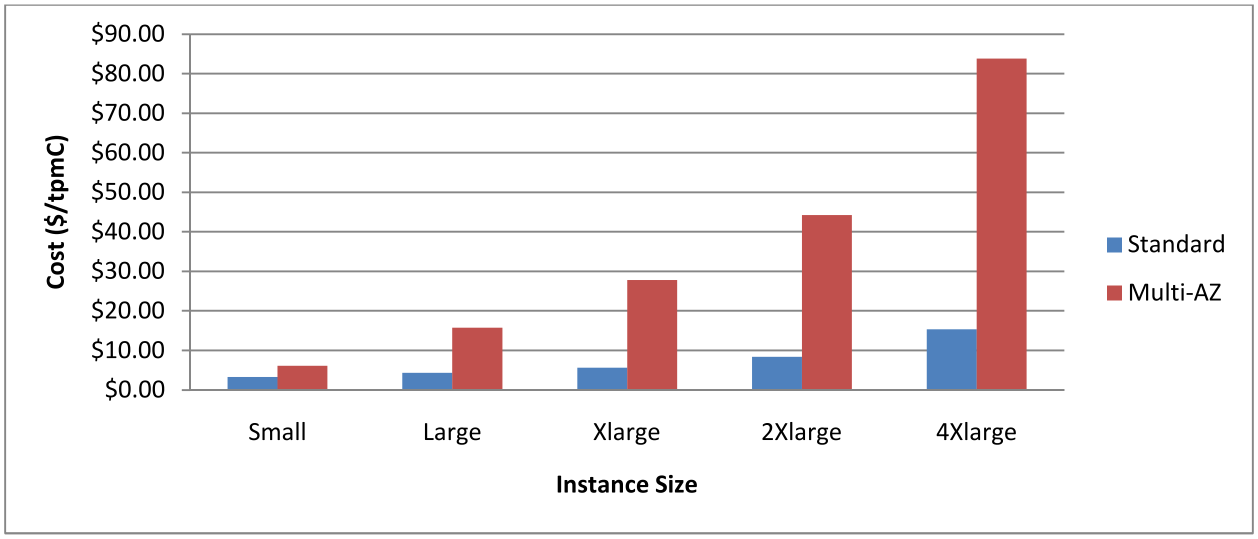


Figure 13: 3-year TPC-C cost ($/tpmC) for Standard and Multi-AZ instances

Summary. In a TPC-C setting it appears standard RDS service features higher throughputs (tpmC) and lower costs ($/tpmC) when compared to Multi-AZ RDS instances. Synchronized back-up to different availability zones causes a high throughput penalty and higher cost per tpmC. For multi-AZ instances, this cost penalty seems to get worse for larger instances. The RDS service throughput increases for larger instances up to a certain load/instance configuration where it peaks. Beyond that, increasing target RDS server instance capabilities does not help. This may be the result of a hidden bottleneck as well as an inherent flaw of the TPC-C benchmark which does not stress the larger instance targets enough to benefit from their additional RAM. Notwithstanding, as a result costs ($/tpmC) decrease up to that threshold peak configuration and then increase with larger instances. A careful application-specific analysis should clarify whether the purchase of larger, more expensive instances is justified. If the application fits the TPC-C benchmark profile the answer seems to be no.