The Impact of Public Cloud Price Schemes on Multi-Tenancy

Uwe Hohenstein, Stefan Appel
Corporate Technology
Siemens AG, Corporate Technology, Otto-Hahn-Ring 6
D-81730 Muenchen, Germany
Email: {Uwe.Hohenstein,Stefan.Appel}@siemens.com

***Abstract*—Multi-tenancy is one key element to make Software-as-a-Service profitable. Multi-tenancy refers to an architecture model where one software instance serves a set of multiple clients of different organizations, i.e., tenants. This reduces the number of application instances and consequently saves operational costs. This paper focuses on using relational databases in multi-tenant architectures, thereby stressing the cost aspect in public cloud environments. Investigating the various price schemes of cloud providers, it illustrates the difficulties to achieve cost-efficient multi-tenancy. As a result, the broad variety of price factors and schemes lead to certain, very different strategies and require adapting multi-tenant architectures to fit the respective cloud providers' specifics.**

Keywords-multi-tenancy; databases; cost; SaaS.

I. INTRODUCTION

Software is more and more becoming an on-demand service drawn from the Internet, known as Software-as-a-Service (SaaS). SaaS is a delivery model that enables customers, the so-called *tenants*, to lease services without local installations and license costs. Tenants benefit from a "happy-go-lucky package": The SaaS vendor takes care of hardware and software installation, administration, and maintenance. Moreover, a tenant can use a service immediately due to a fast and automated provisioning [1].

Multi-tenancy is a software architecture principle allowing SaaS to make full use of the economy of scale: A shared infrastructure for several tenants saves operational cost due to an increased utilization of hardware resources and improved ease of maintenance [4]. Multi-tenancy is often considered as the key to SaaS.

Several authors, e.g., [23], discuss architectures according to what is shared by the tenants: the topmost web frontend, middle tier application server, and underlying database. Concerning the database, [5] describes a number of patterns, which support the implementation of multi-tenancy. We here distinguish between a *1-DB-per-tenant* and a *1-global-DB* strategy. The first one provides a database (DB) of its own for each tenant, thus achieving high data isolation, while several tenants share a common database without physical data isolation in the second variant. Further variants as discussed by [5] are irrelevant in this work.

In this paper, we report on industrial experiences when deploying SaaS in public clouds. Particularly, we focus on cost aspects of multi-tenancy for SaaS using a database because we feel economical aspects not appropriately tackled so far in research. Indeed, economic concerns are important as SaaS providers need to operate with high profit to remain competitive. We here elaborate on the huge differences of price schemes for relational database systems of public cloud providers and the impact on multi-tenancy. Even if various software engineering techniques propose NoSQL databases, relational systems are still often used in industrial applications, especially if being migrated to the Cloud.

Section II presents some related work and motivates why further investigations about cost aspects are necessary. We investigate the price models of various well-known public cloud providers in Section III: Amazon Web Services (AWS), HP Cloud, Microsoft Azure, and Oracle. The price information can be found at their homepages. We discuss in detail the impact of the price models on multi-tenancy strategies and the difficulties to optimize costs. In particular, we quantify the respective costs for implementing multi-tenancy by comparing a 1-DB-per-tenant strategy with a 1-global-DB. Finally, conclusions are drawn in Section IV.

II. RELATED WORK

The work in [4] considers performance isolation of tenants, scalability issues for tenants from different continents, security and data protection, configurability, and data isolation as the main challenges of multi-tenancy. These topics are well investigated. For instance, [17] investigates configurability of multi-tenant applications in case studies.

The possible variants of multi-tenancy have been described, among others, by [5]. Based on the number of tenants, the number of users per tenant, and the amount of data per tenant, [25] makes recommendations on the best multi-tenant variant to use.

Armbrust et al. [1] identify short-term billing as one of the novel features of cloud computing and [8] consider cost as one important research challenge for cloud computing. However, most works on economic issues around cloud computing focus on cost comparisons between cloud and on-premises and lease-or-buy decisions [22]. For example, [9] provides a framework that can be used to compare the costs of using a cloud with an in-house IT infrastructure, and [15] presents a formal mathematical model for the total cost of ownership (TCO) identifying various cost factors. Other authors such as [2][10], focus on deploying scientific applications on Amazon, thereby pointing at major cost drivers. [11] performs the TPC-W benchmark for a Web application with a backend database and compares the costs for operating the web application on several major cloud providers. A comparison of various equivalent architectural

 ISBN: 978-1-61208-460-2

solutions, however, using different components, such as queue and table storage, has been performed by [7]. The results show that the type of architecture can dramatically affect the operational cost.

Cost aspects in the context of multi-tenancy are tackled by [18][19]. They consider approaches to reduce resource consumption as a general cost driver, looking at the infrastructure, middleware and application tier, and what can be shared among tenants.

Another approach, discussed by [24], reduces costs by putting values of utilization and performance models in genetic algorithms.

The authors of [13] develop a method for selecting the best database in which a new tenant should be placed, while keeping the remaining database space as flexible as possible for placing further tenants. Their method reduces overall resource consumptions in multi-tenant environments. Cost factors taken into account are related to on-premises installations: hardware, power, lighting, air conditioning, etc.

Based on existing single-tenant applications, [3] stresses on another cost aspect for multi-tenant applications: maintenance efforts. The recurrence of maintenance tasks (e.g., patches or updates) raises operating cost.

The work in [6] recognizes a viable charging model being crucial for the profitability and sustainability for SaaS providers. Moreover, the costs for redesigning or developing software must not be ignored in SaaS pricing. Accordingly, [18] discusses a cost model for reengineering measures.

The challenges of calculating the costs each tenant generates for a SaaS application in a public cloud are discussed in [21]. This is indispensable to establish a profitable billing model for a SaaS application. The paper shows that only rudimentary support is available by cloud providers.

To sum up, the profitable aspects of multi-tenancy for SaaS providers are researched insufficiently. All the mentioned work is quite general and does mostly not take into account common public cloud platforms and their price schemes. Even best practices of cloud providers, for instance [16] and [19], do not support SaaS providers to reduce cost. As the next section illustrates, there is a strong need to investigate cost aspects for those platforms.

III. COST CONSIDERATIONS

Deploying multi-tenant applications in a public cloud causes expenses for the consumed resources, i.e., the pricing scheme of cloud providers comes into play. Unfortunately, the price schemes for cloud providers differ a lot and are based upon different factors such as the data volume, data transfer, etc. That is why we investigate the price schemes for databases of some major public cloud providers. The goal is to discuss variances in price schemes and how these affect multi-tenancy strategies for SaaS applications. We assume that each tenant demands a certain amount of database storage. We then compare storage that is provided using a dedicated database per tenant with a global database for all tenants to guide a decision.

Please note it is *not* our intention to compare different cloud providers with regard to costs or features. That is the reason why we keep the providers anonymous. There is also no common tool offered by all providers. Furthermore, the price schemes of cloud providers are quite diverging and incorporate different factors. We rather illustrate the variety of price schemes and service offerings leading to different architectures. This also means that the discussion of each offering has a different structure. Moreover, the prices are changing frequently, while the scheme usually remains stable. We here refer to the state as of September 2015.

We only consider resources that are available on-demand to fully benefit from the cloud. This excludes, e.g., reserved instances since those require long-term binding and thus impose a financial risk.

TABLE I. PRICE SCHEME FOR OFFERING 1.

Consumption	*Price*	*Additional GB*
0 to 100 MB	$4.995 (fix price)	
100 MB to 1 GB	$9.99 (fix price)	
1 to 10 GB	$9.99 for 1st 1 GB	$3.996
10 to 50 GB	$45.96 for 1st 10 GB	$1.996
50 to 150 GB	$125.88 for 1st 50 GB	$0.999

A. Offering 1

Offering 1 is a database server available as Platform-as-a-Service (PaaS) in a public cloud. PaaS include licenses and seems to be reasonable for multi-tenancy. Without PaaS, there is no elasticity since licenses must be ordered in time.

Table I presents the recent prices for a Microsoft SQL Server in the US East region. In addition, outgoing data is charged for each database individually with a decreasing rate. The first 5 GB are for free, each additional GB is charged with 8.7ct/GB and 8.3ct/GB above 10 TB, decreasing to 5ct/GB for more than 350TB. However, the cost reduction is insignificant unless there is extremely high outgoing transfer. The storage consumption is the main cost driver. Each database is paid for the amount of stored data. There is no cost difference between using one or several database servers for hosting the databases due to virtualization. We even could not detect any performance difference between placing databases on one virtual server or several ones. One has to pay for the consumed storage in every database – the number of databases and servers is irrelevant.
At a first glance, the price scheme suggests the same costs for 1-DB-per-tenant and 1-global-DB (keeping all tenants). There seems to be no cost benefit for sharing one database between several tenants, since SaaS providers are charged for the total amount of used storage. However, there are indeed higher costs for individual tenant databases since

- sizes larger than 1 GB are rounded up to full GBs;
- smaller databases are more expensive per GB than larger ones due to a progressive reduction.

Since pricing occurs in increments of 1 GB, hundred tenants with each a 1.1 GB database are charged with 100*2 GB, i.e., 100 * $13.986 = $1398.60 a month. In contrast, one database with 100 * 1.1GB = 110 GB is charged with $185.82, i.e., a total difference of $1212.78 or a difference of $12.13 for each tenant (per-tenant difference).

Figure 1 compares the costs of both strategies for various numbers of tenants (10,25,...,400). The x-axis represents the database size, the y-axis the per-tenant difference in US$,

 ISBN: 978-1-61208-460-2

i.e., the additional amount of money a SaaS provider has to pay for *each* tenant compared to a 1-global-DB strategy (note that the prices in Figure 1 must be multiplied by the number of tenants for total costs). The difference stays below $10 for tenant sizes up to 3 GB. The number of tenants is mostly irrelevant. This is why the lines are superposing; only the "10 tenants" line is noticeable. In the worst case, we have to pay $50 more for each tenant with a 1-DB-per-tenant strategy. A linear price drop occurs after 50 GB because even 1-DB-per-tenant uses larger and cheaper databases. Anyway, a 1-DB-per-tenant strategy can become quite expensive compared to a 1-global-DB strategy.

Please note the amount of *used* storage is charged. That is, an empty database is theoretically for free. However, even an empty database stores some administrative data so that the costs are effectively $4.995 per month (for < 100MB). Anyway, these are small starting costs for both a 1-DB-per-tenant and a 1-global-DB strategy.

There is no difference between provisioning a 10 GB and a 150 GB database from a cost point of view as the stored data counts. A 1-global-DB strategy, having the problem not to know how many tenants to serve, can start with 150 GB, thus avoiding the problem of later upgrading databases and possibly risking a downtime while having low upfront cost. Even for a 1-DB- per-tenant strategy, larger databases can be provisioned in order to be able to handle larger tenants without risk.

However, there is a limitation of 150 GB per database, which hinders putting a high amount of tenants with larger storage consumption in a single database. Reaching the limit requires splitting the database into two.

Along with this comes the challenge to determine a cost-efficient placement strategy. Assume an existing 90 GB database and that we need 40 and 30 GB more space for two further tenants: Putting 60 GB into the existing 90 GB database and 10 GB into a new one is the cheapest option with $225.75 + $45.96 = $271.71, more than $70 cheaper than using a new 40 GB and a new 30 GB database: $165.84 + $105.84 + $85.88 = $357.56. Even using a new 70 GB is more expensive with $311.70. An appropriate tenant placement strategy is to fill databases up to the 150 GB threshold, maybe minus some possible space for tenants' expansions, e.g., $286.58 = $205.80 (90+40 GB) + $85.88 (30 GB).

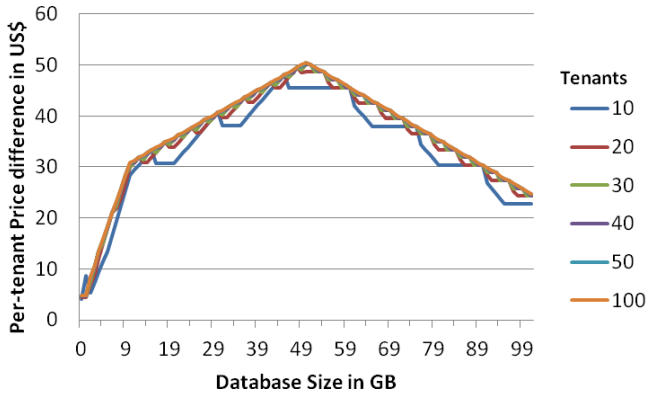

Figure 1. Price difference per tenant for Offering 1.

TABLE II. PRICE SCHEME FOR OFFERING 2.

Level	*Price/month*	*DB size*	*Session limit*	*Transaction rate / hour*
B	~$5	2	300	16,600
S0	~$15	250	600	31,260
S1	~$30	250	900	56,040
S2	~$75	250	1,200	154,200
S3	$150	250		306,000
P1	~$465	500	2,400	378.000
P2	~$930	500	4,800	820,800
P3	~$3,720	500	19,200	2,646,000

B. *Offering 2*

This candidate offers three tiers (**B**asic, **S**tandard, **P**remium). Table II shows the Microsoft SQL Server prices in the US East region for various performance levels inside.

Again, each individual database is paid according to the price scheme. But in contrast to Offering 1, the *provisioning* of the tier is relevant, not the effective storage consumption.

Figure 2 compares per-tenant costs for the 1-DB-per-tenant and 1-global-DB strategies in the same way as in Figure 1. 1-global-DB uses S0 databases, while 1-DB-per-tenant uses B (<= 2 GB) and S0 (> 2 GB) depending on the required size.

One of the worst cases that could happen for 1-DB-per-tenant is to have 100 tenants with 2.2 GB (S0) each, resulting in $1500 per month since each tenant cannot be satisfied with the B tier. In contrast, 1 * 220 GB (S0) for 1-global-DB costs $15. That is a per tenant difference of $14.85. However, it is unclear here whether an S0 level is sufficient for handling 100 tenants from a performance point of view.

A 1-DB-per-tenant strategy is about $5 more expensive if the size is lower than 2GB, and about $15 otherwise. The difference is never higher than $14.77, and drops to $12 for 50 GB and to $9 for 100 GB.

For each database, we have to pay at least $5 a month for at most 2 GB and $15 for up to 250 GB. The costs occur even for an empty database. These baseline costs have to be paid for a 1-gobal-DB, too, starting with the first tenant.

Especially for a 1-global-DB approach, a new challenge arises: Each service level determines not only an upper limit for the database size but also for the number of allowed parallel sessions and the number of (internal) worker threads. Furthermore, there is an impact on the transaction rate (cf. Table II). We have to stay below these limits. Upgrading the category in case of reaching the limit happens online, i.e., without any downtime – in theory: if the database size limit is reached, no further insertions are possible until the upgrade has finished. According to the documentation, such a migration can take several minutes up to hours depending on the database size. If the allowed number of sessions is reached, no further clients can connect unless sessions are released by other users. And if the transaction rate is insufficient, the performance will degrade. Hence, a prediction of tenants' data and usage behavior is required. The number of sessions might become the restrictive factor for a 1-global-DB strategy. In the following, we discuss the impact of the number of users and required sessions on costs by means of sample calculations.

 ISBN: 978-1-61208-460-2

TABLE III. COMPARISON OF CONFIGURATIONS.

	Configuration 1 vs. 2		*# sessions for 1 vs. 2*		*Transaction rate 1000/h (1 vs. 2)*	
a	5*S0	1*S2	3000	1200	156	154
b	2*S0	1*S1	1200	900	62	56
c	2*P1	1*P2	4800	4800	756	820
d	4*P2	1*P3	19200	19200	3283	2646
e	31*S0	1*P1	18600	2400	969	378

Keeping 100 tenants in 1*S0 offers 600 sessions, i.e., 6 sessions per tenant (which might be too small); the monthly costs are $15. We can scale-up to 1*P3 with 19,200 sessions, i.e., 192 per tenant, for a high price of $3720. To achieve the same number of sessions, we can also scale-out to 32*S0 for $480 or use 64*B for $360 if each database is smaller than 2 GB. In contrast, a pure 1-DB-per-tenant strategy for 100 tenants costs $500 for B: This seems to be affordable, especially because of 30,000 sessions. For the price of one P3, we also get 248*S0 databases with 148,000 sessions (6 times more than 1*P3) and a 3 times higher transaction rate of 7,752,480.

For serving 100 tenants with 20 parallel users each, we need 2000 sessions in total. We can achieve this by either 7*B (for $35), 4*S0 ($60), 3*S1 ($90), 1*P1 ($465), or 2*S2 ($500) with very different prices. A pure 1-DB-per-tenant for B is with $500 in the price area of the last two options, but supporting 300 sessions per tenant instead of 20.

Figure 3 illustrates the costs in US$ to achieve x sessions for 100 tenants. B1 represents a pure 1-DB-per-tenant strategy using B-level instances. The P levels are most expensive, even S2 is quite expensive. An obvious question is what the benefit of higher levels in the context of multi-tenancy is. Table III compares several configurations with same prices. There is no consistent behavior. However, several smaller machines seem to be superior to same priced larger ones with a few exceptions. One exception is row (c) where 1*P2 is a little better than 2*P1. More sessions can usually be achieved if n smaller tiers are used instead of one larger one for the same price.

Considering Table II again, we also notice that the session and transaction rates increase from tier to tier less proportional than the prices. Exceptions for transaction rates are S1->S2 and P1->P2. It seems to be reasonable to scale-out instead of scaling-up to obtain more sessions and transactions.

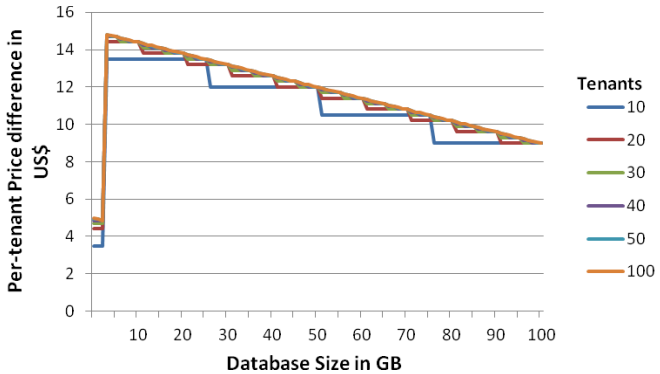

Figure 2. 1-DB-per-tenant vs. 1-global-DB for Offering 2.

TABLE IV. PRICE SCHEME FOR OFFERING 3.

Instance Type	*RAM*	*Storage*	*Price/month*
XS	1 GB	15 GB	$73
S	2 GB	30 GB	$146
M	4 GB	60 GB	$292
L	8 GB	120 GB	$584
XL	16 GB	240 GB	$1,168
XXL	32 GB	480 GB	$2,336

Another advantage is that baseline costs can be saved. A 1-global-DB strategy requires a high-level database with a high price already for the first tenant independent of the number of eventually stored tenants.

Indeed, it is difficult to derive a strategy for identifying a suitable configuration. Important questions arise:

- Is an upgrade possible in short time, without outage? This would allow for 1-global-DB to start small for few tenants and upgrade if performance suffers or the number of sessions increases. For 1-DB-per-tenant, we could start with B and upgrade to S0.
- Are only the session and transaction rates of a level relevant, or are there any other (invisible) performance metrics to consider? The documentation mentions only that the predictability, i.e., the consistency of response times, is increasing from B to Px, however being the same within a tier.

C. Offering 3

Offering 3 provides a MySQL database as PaaS. The regular prices are for virtualized databases on an hourly basis. The payment is based upon the following factors:

- The instance type, which limits the maximal database size and determines the RAM (cf. Table IV).
- The outgoing data transfer: the first GB is for free, we then pay 12ct/GB up to 10 TB, further GBs for 9ct up to 40 TB, etc.

In contrast to Offering 1, the provisioned storage is paid. The prices and the features increase with the instance type linearly, i.e., each next higher instance type doubles the RAM and maximal database size for a doubled price.

Comparing the strategies, we notice that 5 tenant databases à 15 GB (XS) are charged with $365. One global database à 75 GB is more expensive (!) with $584 since we are forced to provision a 120GB (L) database. The difference per tenant is $43.80. However, using 15GB (XS) increments for 1-global-DB, we can achieve the cheaper price.

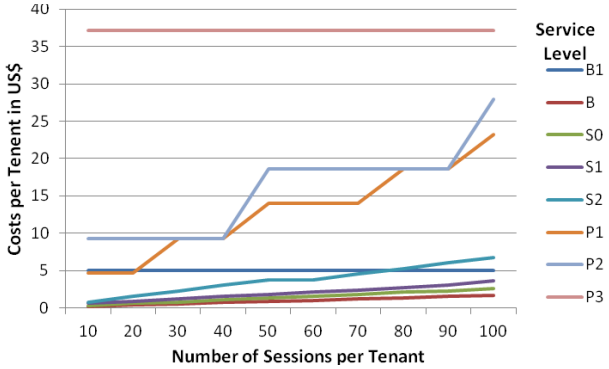

Figure 3. Costs to achieve x sessions per tenant for Offering 2.

TABLE V. PRICE SCHEME FOR OFFERING 4.

Type	*Max database size*	*Data Transfer*	*Price/month*
5GB	5 GB	30 GB	$175
20GB	20 GB	120 GB	$900
50GB	50 GB	300 GB	$2,000

Hence, we should use XS partitions in order not to pay for unused storage. Thus, an appropriate cost strategy for 1-global-DB is to fill XS databases one by one with tenants. However, this has architectural implications in order to connect each tenant to the right database instances. A 1-DB-per-tenant approach could also benefit that way. There is no need to use larger instances unless we do not want to spread tenant data across databases due to implementation effort.

A worst case scenario is storing 15 tenants with 100MB each. 1-DB-per-tenant is charged with $1095 = (15*XS), while one global XS database costs $73 for 1.5 GB. That is a difference per tenant of $68.13.

Figure 4 illustrates that 1-DB-per-tenant, compared to 1-global-DB based upon XS databases, is more expensive for sizes much smaller than the storage threshold. Reaching the threshold, the difference diminishes. Hence, it is reasonable to use one database for each tenant if the storage size is near a threshold. In summary, we observe larger per-tenant differences depending on database sizes. The range where the difference stays below $20 is very small. Moreover, the variances for different numbers of tenants are small.

An incremental acquisition of XS databases even saves baseline costs. However, it is an open issue to be investigated whether larger instances provide a better performance. The time for upgrading from one instance type to another is not important here.

D. Offering 4

Three database types are available, each limiting the maximal amount of storage. Table V shows that each type also limits the allowed data transfer.

A comparison of the types gives some first insights: 20GB is 5 times more expensive than 5GB, but offers only 4 times more data transfer and storage. 50GB is 2.2 times more expensive than 20GB, but offers 2.5 times more data transfer and storage. And 50GB is 11 times more expensive than 5GB, but offers only 10 times more resources. Hence, 20GB has the worst price ratio, 5GB the best one. Obviously, using 5GB databases seems to be reasonable for either strategy unless we do not want to spread tenant data across databases.

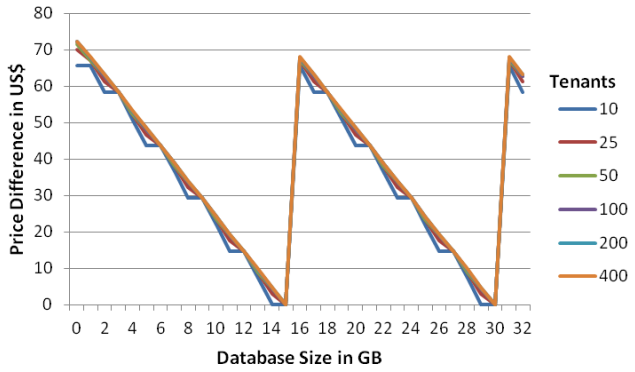

Figure 4. Price diff. of 1-DB-per-tenant vs. 1-global-DB for Offering 3.

TABLE VI. COMPARISON OF SAMPLE CONFIGURATIONS

Config	*#tenants*	*database size*	*Costs*	*Data transfer*	*Per-tenant costs*
100*5GB	100	1 GB	17,500	3000	175
2*50GB			4,000	600	40
5*20GB			4,500	600	45
20*5GB			3,500	600	35
200*5GB	200	4 GB	35,000	6000	175
16*50GB			32,000	4800	160
40*20GB			36,000	4800	180
100*5GB	100	5 GB	17,500	3000	175
10*50GB			20,000	3000	200
25*20GB			22,500	3000	225

Table VI compares a 1-DB-per-tenant configuration (the first lines) with others. For 200 tenants à 4GB, using 20GB databases is more expensive than 1-DB-per-tenant; the same holds for 100 tenants à 5GB.

Figure 5 summarizes the price-per-tenant differences if 5GB increments are used. A 1-DB-per-tenant strategy is only reasonable if the database size is near a multiple of 5 GB, or if the required data transfer is high. The larger the distance is, the higher will be the per-tenant costs compared to a 1-global-DB. This saw tooth behavior is repeating. The number of tenants has again no impact.

Since the data transfer is limited by the instance type, a challenge arises for the 1-global-DB strategy: this can stop several or all tenants from accessing the database. Additional data transfer cannot be acquired even for extra charges.

A possible strategy for 1-global-DB is to start with 5GB and to add further ones later; this means less upfront costs. Moreover, 5GB is the cheapest category wrt. gains. Please note that downsizing is not possible. This causes further costs in case a tenant stops using the SaaS service.

E. Offering 5

Offering 5 provides a virtual machine (VM) with a Microsoft SQL Server for various operating systems. The price model is quite complex covering several factors.

At first, a VM has to be chosen for hosting the database server. Table VII summarizes the prices for a Windows OS in the East US region. Each tier has a different number of virtual cores (vCores), RAM, and temporary disk space. A0-A7 covers the standard tier; A0-A4 are also available in a basic tier with little lower prices ($13-$440) than the standard tier.

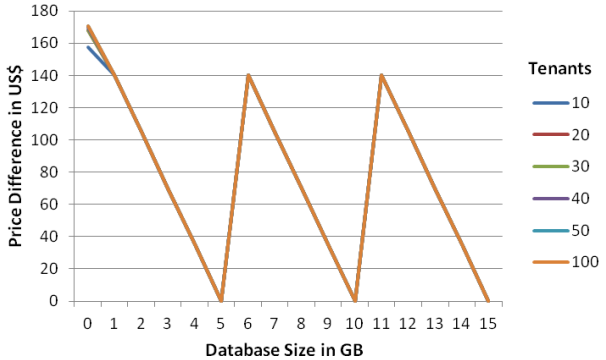

Figure 5. Price diff. of 1-DB-per-tenant vs. 1-global-DB for Offering 5.

TABLE VII. PRICE SCHEME FOR OFFERING 5

Tier	*vCores*	*RAM*	*TempDisk*	*price/month*	*#disks*
A0	1	768 MB	20 GB	$15	1
A1	1	1.75 GB	70 GB	$67	2
A2	2	3.5 GB	135 GB	$134	4
A3	4	7 GB	285 GB	$268	8
A4	8	14 GB	605 GB	$536	16
A5	2	14 GB	135 GB	$246	4
A6	4	28 GB	285 GB	$492	8
A7	8	56 GB	605 GB	$983	16
A8	8	56 GB	382 GB	$1,823	16
A9	16	112 GB	382 GB	$3,646	16
D1	1	3.5 GB	50 GB	$127	1
D2	2	7 GB	100 GB	$254	2
D3	4	14 GB	200 GB	$509	4
D4	8	28 GB	400 GB	$1,018	8
D11	2	14 GB	100 GB	$600	2
D12	4	28 GB	200 GB	$1,080	4
D13	8	56 GB	400 GB	$1,943	8
D14	16	112 GB	800 GB	$2,611	15

A8 and A9 are network-optimized instances adding an InfiniBand network with remote direct memory access (RDMA) technology. The D-tier is compute-optimized with 60% faster CPUs, more memory, and a local SSD. An OS disk space of 127 GB is available and must be paid with ignorable 2.4ct per GB/month.

Furthermore, the server is charged per minute. The prices depend on the number of cores of the used VM: $298 for 1-4 core machines, $595 for 8 cores, and $1190 for 16 cores for a SQL Server Standard Edition in a month. The Enterprise Edition is more expensive, e.g., $4464 for a 16-core VM.

Additional costs occur for attached storage. There is a maximum of number of 1TB data disks (#disks in Table VII). The costs can be neglected with 5ct/GB-month for the first 1000 TB of consumed storage in a page blob. The costs are thus dominated by other factors than disk space.

As a major difference to previous price schemes, a database *server* is provisioned and paid instead of a single database. The database server offers full control like operated on-premises. Several databases can be managed in that server. This directly implies that a 1-DB-per-tenant strategy is feasible, i.e., each tenant can obtain a database of its own with individual credentials. A strong isolation is thus given without any extra charge. As a consequence, a strategy could be to use one database server and set up one database for each tenant until performance decreases.

Instead of comparing 1-DB-per-tenant and 1-global-DB, we have to consider how many database servers (hosting several databases) of what tier we have to apply for the expected number of tenants and users. One strategy could be to start with a small VM and increase the instance type with the number of tenants. This implies that such an upgrade is possible within short time.

If an upgrade could cause a downtime, we have to decide whether to use several small VMs or few larger ones from a cost perspective. There are high minimal costs of at least $311 per month for each database server ($13 for the smallest Windows VM A0 Basic plus the database server). A high number of tenants/databases will obviously require larger VMs, leading to higher baseline costs.

TABLE VIII. CONFIGURATIONS TO ACHIEVE 112 GB RAM.

configuration	*# vCores*	*Price (decreasing)*
8*D11	16	$4,800
8*A4	64	$4,288
8*D3	32	$4,072
2*D13	16	$3,886
A9	16	$3,646
2*A8	16	$3,646
D14	16	$2,611
8*A5	16	$1,968
2*A7	16	$1,966

The question is what configuration is sufficient for a given number of tenants and amounts of data. Unfortunately, no performance hints are provided to ease the decision. It might be better to provision a larger VM since it can be used for other purposes as well if being idle. A brief evaluation of an SQL Server Standard Edition shows the following:

a) 1*A9 costs $4836 ($3646 for the VM, $1190 for the SQL Server), offering 16 vCores and 112 GB RAM.

b) For the same price we get 8.5*A3 with ~34 vCores and ~60 GB RAM in total.

c) Alternatively, we can also purchase 3*A7 with 24 vCores and 168 GB RAM.

d) 13*A1 comes for $4745 with ~23 GB RAM and 13 vCores.

(d) offers the least equipment for the price because of the high number of database servers, each for $298. Options (b) and (c) favor either vCores or RAM. In general it looks reasonable to avoid high-class VMs and to use several middle-class VMs.

An incremental provisioning can also help to reduce upfront baseline costs, which occur already with the first tenant. However, a deeper empirical performance evaluation is necessary due to further open questions:

- What VM should be chosen? Table VIII shows several options to achieve 112 GB RAM with very different prices and numbers of vCores.
- Similarly, there are many variants for 14 or 28 GB RAM, each yielding different vCores with prices ranging from $246 to $600 for 14GB RAM, and from $492 to $1089 for 28 GB RAM.
- Finally, when pursuing the approach with one or few small servers, it is indispensable to know how long it last to upgrade the category of a VM.

F. Offering 6

Similar to Offering 5, this option again offers a VM running a database server, however, with several differences regarding pricing. Several database systems such as MySQL, Oracle, PostgreSQL, and SQL Server are supported in various database instance classes of three categories: Micro, Standard, and Memory-optimized.

The prices depend on the chosen instance type, the type of database server, and the region. The prices for a MySQL database in the US East region are presented in Table IX. The underlying VM and the MySQL license are already included in the price. The instance class determines the number of virtual CPUs (vCPU) and the main memory.

 ISBN: 978-1-61208-460-2

TABLE IX. PRICE SCHEME FOR OFFERING 6.

Category	*Instance type*	*Price / month*	*vCPU*	*RAM*
Micro	XS	$12.41	1	1
	S	$24.82	1	2
	M	$49.64	2	4
Standard	M	$65.70	1	3.75
	L	$135.05	2	7.5
	XL	$260.10	4	15
	XXL	$520.20	8	30
Memory-optimized (MemOpt)	L	$175.20	2	15
	XL	$346.75	4	30.5
	XXL	$689.85	8	61
	4XL	$1379.70	16	122
	8XL	$2759.40	32	244

An additional cost factor is the outgoing data transfer to the Internet of 9ct/GB: prices decrease with the amount, 1 GB-month is for free. The price decrease for larger volumes is insignificant.

Furthermore, the amount of data is charged according to two alternative classes of database storage:

a) General purpose for 11.5ct per GB-month with a range from 5 GB to 3 TB; 3 IOPS per GB are included.

b) Provisioned IOPS for 12.5ct per GB-month and additional $0.10 per requested IOPS/month, with a range from 100 GB to 3 TB and 1,000 IOPS to 30,000 IOPS.

IOPS (IO per second) determines an upper limit for IO. IO itself is not charged.

Again, we have to decide how many servers are reasonable. The baseline costs for each database server are determined by the minimal settings: The smallest installation in terms of cost for MySQL is Micro XS with $12.41/month. Provisioned IOPS storage is available at a minimum of 100 GB and 1000 IOPS, i.e., $12.50 (100*12.5ct) plus $100 (1000 IOPS à 10ct) ending up with costs of at least $112.50 per server. Using alternate general purpose storage, we have to provision at least 5 GB for 57.5ct (5*11.5ct); but then only 15 IOPS are available (see (a) above). Hence, setting up a minimal MySQL server, e.g., for each tenant, comes with at least $13 using general purpose storage, while provisioned storage is much more expensive with $125.

A calculation and comparison of using one or several database servers is difficult since several factors are unclear. A high-end server might be more appropriate since the high provisioning cost for storage occur only once. According to the documentation, the network performance also increases with a higher instance class. However, many smaller servers avoid higher, tenant-independent upfront investments for a larger instance and required IOPS.

Table X shows some configurations with similar monthly costs. The provided equipment differs a lot. Obviously, (a) is better equipped than (b). But each of (a) and (d) has an advantage for vCPUs or RAM, respectively.

It is important to note that the *provisioned* numbers are relevant, not the effective usage. This means, the required storage for each tenant has to be estimated in order not to overpay for unused resources. The same holds for the IOPS rate. These costs occur already for the first tenant independently of consumed resources.

TABLE X. COMPARISON OF CONFIGURATIONS.

	Configuration	*vCPUs*	*RAM*	*Costs*
a	1* 8XL (mem-opt)	32	244	$2759
b	2 * 4XL (standard)	16	244	$2759
c	20 * L (standard)	40	150	$2704
d	42 * M (standard)	42	157	$2759

One strategy could be to start with small servers and increase the instance type and configuration if necessary. This implies that such an upgrade is possible within short time and without downtime in the meantime. Otherwise, a larger machine with larger storage and IOPS can be provisioned from the beginning, however, causing high starting costs already for some few tenants.

Another strategy is to use one smaller server for each tenant, e.g., for $13. Then, the expenses increase tenant by tenant. This also gives more flexibility for provisioning IOPS according to tenants' requirements.

An important question is what IOPS rate is sufficient since the IOPS rate is a limiting factor: Throttling of users can occur if the limit is reached. Obviously, keeping several tenants in one server requires higher IOPS rates. It is unclear what the advantage of provisioned storage is compared to general purpose storage. From a pure cost perspective, 6000 IOPS are charged with $600 for provisioned storage. To achieve 6000 IOPS with general purpose storage, we have to use 2TB (remind the factor in (a)), i.e., being much cheaper with $230 and already including storage. Since there is an upper database limit of 3 TB in any case, general purpose IOPS ends with 9,000 IOPS; provisioned storage can handle up to 30,000 IOPS.

IV. CONCLUSIONS

This paper took a deeper look into the price schemes of popular cloud database providers and investigated their cost impact on multi-tenancy. We thereby focused on storing tenants' data in relational databases. We showed that a cost-efficient database deployment for multi-tenancy heavily depends on providers due to very different price schemes. Several differences become apparent.

- Offering 2-6 charge for provisioned storage, i.e., upfront costs occur even if small data is stored. In contrast, Offering 1 charge for storage consumption which avoids starting costs instead.
- Sometimes, databases are paid (cf. Offerings 1-4); sometimes whole DB servers are provisioned (cf. Offering 5 and 6) so that several isolated databases can be managed with specific credentials.
- Offerings 2 and 4 define certain limits on transaction rates, data transfer, or number of sessions. Reaching such a limit could stop a SaaS application for serving tenants.
- For Offerings 3, 5, and 6, equipment such as RAM increases with each level, while this is not controllable and visible in Offerings 1, 2, and 4.

There are direct cost factors such as storage, IOPS, sessions, cores, or data transfer, i.e., they are directly part of the price scheme. We detected indirect cost factors, too. For example, it might be necessary to use and pay a larger virtual

machine (VM) in order to achieve a certain transaction rate, e.g., Offering 2.

The broad spectrum of price schemes makes it difficult to find an appropriate provider-independent cost-optimized configuration for multi-tenant applications. However, we could present some analyses comparing the cost of a 1-DB-per-tenant and a 1-global-DB strategy and displaying the characteristics for different tenant sizes. The results also have a strong impact on the cloud provider selection. For example, if a strong isolation is requested, a provider with too high prices for a 1 DB-per-tenant strategy might not be qualified for a selection.

As a consequence, it is difficult to select *the* best provider from the cost perspective. But we think that our analysis helps architects of multi-tenant software to decide upon a cloud offering for the anticipated requirements. Besides architects, cloud providers can benefit from our analysis when it comes to adjust their service offerings.

This all affects portability of SaaS applications, too. It is not easy to define an economic provider-independent strategy for multi-tenancy. Furthermore, architectures must take into account several aspects. For example, monitoring consumption becomes necessary because of thresholds such as a database upper limit of parallel sessions, IO limits, or any other type of throttling. This is indispensable to react in time if a threshold is reached because a service is in danger of being stopped.

Future work will tackle open questions, including practical investigations. One important question is about the provisioning time. This point is relevant in any strategy since additional databases have to be acquired. Similarly, upgrading a database level is important for saving upfront costs.

Finally, we intend to collect further challenges from an industrial perspective.

REFERENCES

[1] M. Armbrust et al., "A View of Cloud Computing," Communications of the ACM, 53(4), April 2010, pp. 50-58.

[2] B. Berriman, G. Juve, E. Deelman, M. Regelson, and P. Plavchan, "The Application of Cloud Computing to Astronomy: A Study of Cost and Performance," Proc. of 6th IEEE Int. Conf. on e-Science, 2010, pp. 1-7.

[3] C. Bezemer, A. Zaidman, B. Platzbeecke, T. Hurkmans, and A. Hart, "Enabling Multitenancy: An Industrial Experience Report," in: Technical Report of Delft Uni. of Technology, TUD-SERG-2010-030, 2010.

[4] C. Bezemer and A. Zaidman, "Challenges of Reengineering into Multitenant SaaS Applications," in: Technical Report of Delft Uni. of Technology, TUD-SERG-2010-012, 2010.

[5] F. Chong, G. Carraro, and R, Wolter, "Multi-Tenant Data Architecture," June 2006, http://msdn.microsoft.com/en-us /library/aa479086.aspx [retrieved: February 2016]

[6] T. Dillon, C. Wu, and E. Chang, "Cloud Computing: Issues and Challenges," in Proc. 24th Int. Conf. on Advanced Information Networking and Applications, 2010, pp. 27-33.

[7] U. Hohenstein, R. Krummenacher, L. Mittermeier, and S. Dippl, "Choosing the Right Cloud Architecture - A Cost Perspective," in Proc. on Cloud Computing and Services Science (CLOSER), 2012, pp. 334-344.

[8] A. Khajeh-Hosseini, I. Sommerville, and I. Sriram, "Research Challenges for Enterprise Cloud Computing," in Proc. 1st ACM Symposium on Cloud Computing, SOCC 2010, Indianapolis, pp. 450-457.

[9] M. Klems, J. Nimis, and S. Tai, "Do Clouds Compute? A Framework for Estimating the Value of Cloud Computing," in Designing E-Business Systems. Markets, Services, and Networks, Lecture Notes in Business Information Processing, Vol. 22, 2008, pp.110-123.

[10] D. Kondo, B. Javadi, P. Malecot, F. Cappello, and D. Anderson, "Cost-Benefit Analysis of Cloud Computing versus Desktop Grids," in Proc. of the 2009 IEEE Int. Symp. on Parallel&Distributed Processing, May 2009, pp 1-12.

[11] D. Kossmann, T. Kraska, and S. Loesing, "An Evaluation of Alternative Architectures for Transaction in Processing in the Cloud," ACM SIGMOD 2010, pp. 579-590.

[12] R. Krebs, C. Momm, and S. Kounev, "Architectural Concerns in Multi-Tenant SaaS Applications," in CLOSER 2012, pp. 426-431.

[13] T. Kwok and A. Mohindra, "Resource Calculations With Constraints, and Placement of Tenants and Instances for Multi-Tenant SaaS Application," in Proc. Int. Conf. on Service-Oriented Computing, (ICSOC) 2008. LNCS, vol. 5364, pp. 633-648.

[14] M. Lindner, F. Galán, and C. Chapman, "The Cloud Supply Chain: A Framework for Information, Monitoring, Accounting and Billing," in Proc. on ICST Cloud Computing, 2010, pp. 1-22.

[15] B. Martens, M., Walterbusch, and F. Teuteberg, "Evaluating Cloud Computing Services from a Total Cost of Ownership Perspective," 45th Hawaii International Conference on System Sciences (HICSS-45), 2012, pp. 1564-1572.

[16] Microsoft, "Developing Multitenant Applications on Windows Azure". http://msdn.microsoft.com/en-us/library /ff966499.aspx [retrieved: January 2016]

[17] R. Mietzner, F. Leymann, and M. Papazoglou, "Defining Composite Configurable SaaS Application Packages Using SCA, Variability Descriptors and Multi-Tenancy Patterns," in 3rd Int. Conf. on Internet and Web Applications and Services (ICIW), 2008, pp. 156-161

[18] C. Momm and R. Krebs, "A Qualitative Discussion of Different Approaches for Implementing Multi-Tenant SaaS Offerings," in Proc. Software Engineering 2011, pp. 139-150.

[19] C. Osipov, G. Goldszmidt, M. Taylor, and I. Poddar, "Develop and Deploy Multi-Tenant Web-Delivered Solutions Using IBM Middleware: Part 2: Approaches for Enabling Multi-Tenancy," in: IBM's technical library, 2009.

[20] A. Schwanengel, U. Hohenstein, and M. Jaeger, "Automated Load Adaptation for Cloud Environments in Regard of Cost Models," in Proc. on CLOSER, 2012, pp. 562-567.

[21] A. Schwanengel and U. Hohenstein, "Challenges with Tenant-Specific Cost Determination in Multi-Tenant Applications," in 4th Int. Conf. on Cloud Computing, Grids and Virtualization, Valencia (2013), pp. 36-42.

[22] E. Walker, "The Real Cost of a CPU Hour," Computer 2009, Vol. 42(4), pp. 35-41.

[23] S. Walraven, E. Truyen, and W. Joosen, "A Middleware Layer for Flexible and Cost-Efficient Multi-Tenant Applications," in Proc. on Middleware, 2011 (LNCS 7049), pp. 370-389.

[24] D. Westermann and C. Momm, "Using Software Performance Curves for Dependable and Cost-Efficient Service Hosting," in Proc. on Quality of Service-Oriented Software Systems (QUASOSS), 2010, pp. 1-6.

[25] Z. Wang et al, "A Study and Performance Evaluation of the Multi-Tenant Data Tier Design Pattern for Service Oriented Computing," in IEEE Int. Conf. On eBusiness Engineering, (ICEBE) 2008, 94-101

 ISBN: 978-1-61208-460-2