Cost Considerations About Multi-Tenancy in Public Clouds

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

Abstract— **Multi-tenancy is often considered as the key element to economical Software-as-a-Service as it represents an architecture model where one software instance serves a set of multiple clients of different organizations – the so-called tenants. This reduces operational costs due to the decrease of the number of application instances and required resources. Since nearly every multi-tenant system requires a database, this paper focuses on database aspects of multi-tenancy and particularly stresses on cost aspects in public cloud environments. Starting with an investigation of the price schemes of cloud providers, this paper illustrates the broad variety of price factors and schemes. It is discussed in detail why this makes it difficult to set up vendor-independent strategies to achieve cost-efficient multi-tenancy and what challenges arise. Anyway, the paper derives certain, vendor-specific strategies, which 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.

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. Thus, multi-tenancy is often considered as the key to SaaS.

Several authors discuss architectures according to what is shared by tenants: the topmost web frontend, middle tier application server, and underlying database. Concerning the database, a number of patterns exist, which support the implementation of multi-tenancy.

This paper extends the work presented in [1]. We report on industrial experiences when deploying multi-tenant SaaS in public clouds. Particularly, we focus on cost aspects of multi-tenancy for SaaS 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. This is challenging due to diverging price schemes. Since nearly every multi-tenant system requires a database, we focus on database aspects of 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. We elaborate on the huge differences of price schemes for relational database systems of public cloud providers and discuss the impact on multi-tenancy.

Compared to our previous work in [1], we here update the price schemes and take into account two further Cloud offerings. Comparing both papers thus illustrate how prices and price schemes evolve over time. Moreover, we applied a uniform and systematic analysis for all Cloud database offerings.

Section II presents some related work and motivates why further investigations about cost aspects are necessary. We introduce general cost aspects in Section III before discussing the price models of various well-known public cloud providers in two broad categories: Virtual databases in Section IV and virtual database servers in Section V. The particular offerings are from Amazon Web Services (AWS), HP Cloud, IBM, Microsoft Azure, and Oracle. 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* and a *1-global-DB* strategy. The first one provides a virtual database (DB) of its own for each tenant, thus achieving high data isolation. In the second variant, several tenants share a common database without physical data isolation. Since some offerings provide a virtual database server (instead of a single database), the same distinction can be made for DB servers. Further variants as discussed by [2] are irrelevant in this work. For Virtual Machines (VMs) with a database server, we investigate scale-out scenarios. Section VI summarizes some major findings. Finally, conclusions are drawn in Section VII.

II. STATE OF THE ART

The work in [3] 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, [4] discusses configurability of multi-tenant applications in case studies.

The possible variants of multi-tenancy have been described, among others, by [2][5][6]. Based on the number of tenants, the number of users per tenant, and the amount of data per tenant, [7] makes recommendations on the best multi-tenant variant to use.

Armbrust et al. [8] identify short-term billing as one of the novel features of cloud computing and [9] consider cost as one important research challenge for cloud computing. However, cost aspects do need seem to be a recent research topic since most work is about 5 years old. Moreover, existing work on economic issues around cloud computing mostly focus on cost comparisons between cloud and on-premises and lease-or-buy decisions [10]. For example, [11] provides a framework that can be used to compare the costs of using a cloud with an in-house IT infrastructure, and [12] presents a formal mathematical model for the total cost of ownership (TCO) identifying various cost factors. Other authors such as [9][13], focus on deploying scientific applications on Amazon, thereby pointing at major cost drivers. [14] 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 solutions, however, using different components, such as queue and table storage, has been performed by [15]. 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 [16][17]. 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 [18], reduces costs by putting values of utilization and performance models in genetic algorithms.

The authors of [19] 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, [20] stresses on another cost aspect for multi-tenant applications: maintenance efforts. The recurrence of maintenance tasks (e.g., patches or updates) raises operating cost as well.

The work in [21] 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, [16] 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 [22]. 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 [17] and [23], 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.

Please note it is *not* our intention to compare different cloud providers with regard to costs or features: For example, The Oracle cloud offers Oracle database servers, which are not provided by the Microsoft cloud, and vice versa. Hence, there is no common basis for a direct comparison of providers. This is also the reason why we keep the providers anonymous. 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 strategies. Moreover, the prices are changing frequently, while the scheme usually remains stable. We here refer to the state as of August 2016. The price information can be found at their homepages.

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.

To structure the discussion, we distinguish between two major categories in Section IV and V, virtual databases and virtual database servers, respectively. While a virtual database server offers full control like operated on premises, particularly allowing one to create several databases within that server, a virtual database is just one database managed by the provider without further control.

IV. Virtual Databases

In this section, we assume that each tenant demands a certain amount of database storage. We then compare the costs for storage that is provided using a dedicated database per tenant (1-DB-per-tenant) with a global database for all tenants (1-global-DB) to guide a decision. Please note the term “database” is used in the sense of something that can be authenticated individually.

A. *Offering 1*

Offering 1 is a database server available as Platform-as-a-Service (PaaS) in a public cloud. PaaS is one of three delivery models according to the NIST definition [24]: “The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.“ That is, a database can be provisioned without any installation and administration. PaaS performs an automatic management (patches etc.) and internal replication of data. In particular, PaaS includes software licenses, which seems to be reasonable for multi-tenancy. There will be no elasticity without PaaS, since licenses must be ordered in time.

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

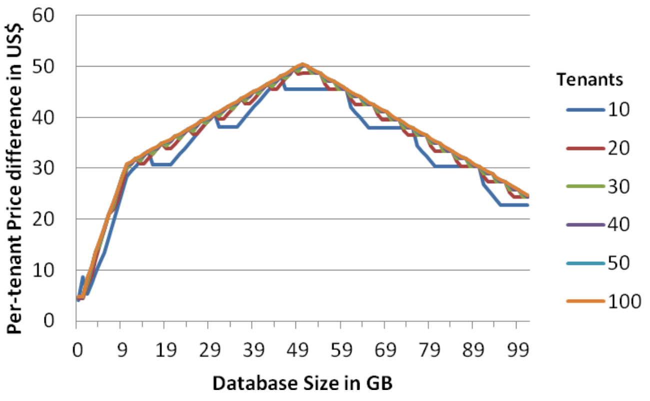

Figure 1. Price difference per tenant for Offering 1.

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. There is only a 40ct benefit for 100GB of outgoing data.

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 independently of how many databases and servers are used.

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 amongst 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 (10,20,...,100). The x-axis represents the database size, the y-axis the per-tenant difference in US$, 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 to obtain the 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).

Please note this offering has been deprecated and replaced with Offering 2 by the cloud provider since our last analysis in [1].

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.

TABLE II. PRICE SCHEME FOR OFFERING 2.

Level	*Price/ month*	*DB size*	*Session limit*	*Max. concurrent logins & requests*	*Transaction rate / hour*
B	~$5	2	300	30	20,160
S0	~$15	250	600	60	31,320
S1	~$30	250	900	90	61,200
S2	~$75	250	1,200	120	165,600
S3	$150	250	2400	200	367,200
P1	~$465	500	2,400	200	457.200
P2	~$930	500	4,800	400	946,800
P4	~$1,860	500	9,600	800	1,778,400
P6	~$3,720	500	19,200	1,600	3,988,800
P11	~$7,001	1000	32,000	3,200	6,339,600

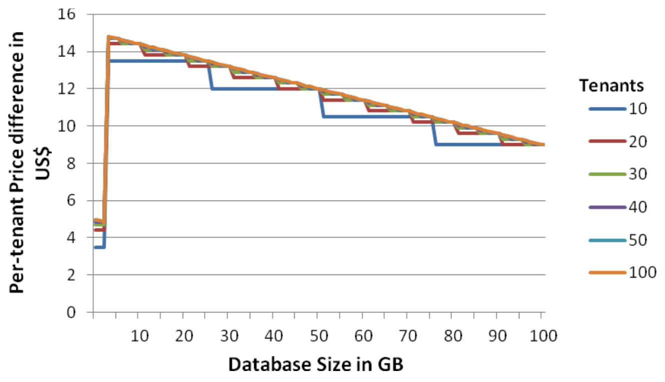

Figure 2. 1-DB-per-tenant vs. 1-global-DB for Offering 2.

Again, this is a PaaS offering for a virtual database. 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. We here use S0 databases for 1-global-DB because S0 are the cheapest option: To store 50GB each for 20 tenants, we can use 4*S0 ($60), 2*P1 ($930) or 1*P11 ($7,001). But it is important to note that the global DB is then distributed over several databases. 1-DB-per-tenant uses B (<= 2 GB) and S0 (> 2 GB) depending on the required size. It becomes obvious that a global database with 2*P1 (with $930) is even more expensive than 1-DB-per-tenant with 20*S0 ($300).

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 2 GB, 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 upfront 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 concurrent requests. 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 our experiences, 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.

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 available 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 within each group 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.

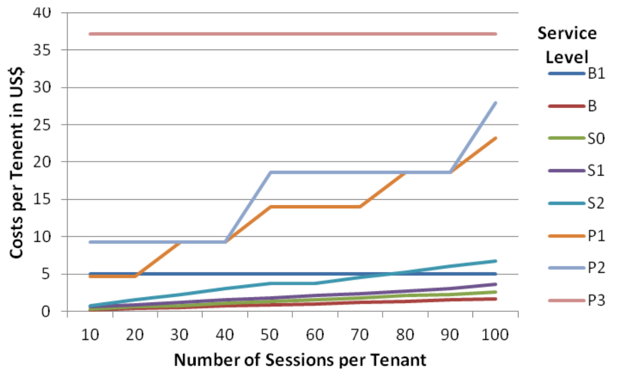

Figure 3. Costs to achieve x sessions per tenant for Offering 2.

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

Another advantage is that upfront 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.

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

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 120 GB (L) database. The difference per tenant is $43.80. However, using 15 GB (XS) increments for 1-global-DB, we can achieve the cheaper price.

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. However, an implication is that additional administrative effort (e.g., for backup) becomes necessary, and even more, the application's code is affected by distributing and collecting data from several XS databases in case of customer data "fragmentation". This has an impact on the total cost (cf. [3]) but is beyond the scope of this paper.

A worst case scenario is storing 15 tenants with 100 MB 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.

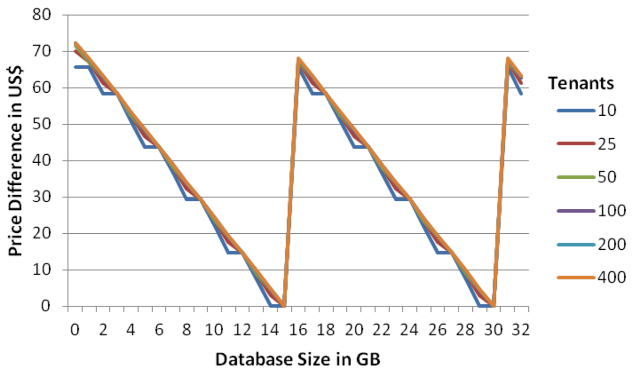

Figure 4. Price diff. of 1-DB-per-tenant vs. 1-global-DB for Offering 3.

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. Note, Figure 4 has a different scale compared to Figure 1 and 2, but the saw tool behavior is repetitive up to 100 tenants and beyond.

An incremental acquisition of XS databases even saves upfront 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 in this PaaS offering, each limiting the maximal amount of storage. Table V shows that each type also limits the allowed data transfer.

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

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.

Table VI compares a 1-DB-per-tenant configuration (the first line in each group) 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.

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

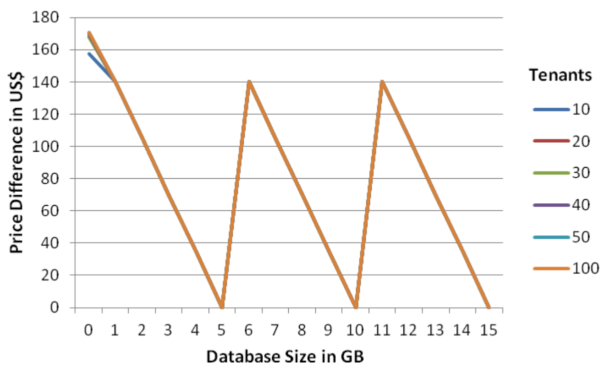

Figure 5. Price Diff. of 1-DB-per-tenant vs. 1-global-DB for Offering 4.

Figure 5 summarizes the price-per-tenant differences if 5GB increments are used in both strategies. 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 real impact. The per-tenant costs can be even higher in this offering than for Offering 3.

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 also note that downsizing is not possible. This causes further costs in case a tenant stops using the SaaS service.

V. VIRTUAL DATABASE SERVERS

As a major difference to the offerings in the previous section, in this category a virtual machine (VM) with 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, especially one dedicated database for each tenant with individual credentials. This directly implies that a 1-DB-per-tenant strategy is feasible. A strong isolation is thus given without any extra charge. As a consequence, the question is not about 1-global-DB vs. 1-DB-per-Tenant, but instead how many database servers with what equipment we have to apply for the expected number of tenants and users. To perform an evaluation, we assume that the number of tenants, the amount of data, and the access profiles are known. The question is what VM configuration is sufficient for a given number of tenants and amounts of data. This allows for a better comparison of equipments for given prices.

Unfortunately, the tenants are coming time after time. This means that the *upfront costs* for the cheapest database server are important, since these are the starting costs.

Moreover, we investigate how the costs evolve with the number of tenants. There are basically two strategies: The first *scale-up* strategy uses a cheap database server and sets up another database within this server for each new tenant until the overall performance decreases. Whenever the existing server reaches performance limits, the type of server is upgraded to a higher tier being better equipped; a high number of tenants/databases can be handled by a larger VM. However, this implies that such an upgrade must be possible within short time and without downtime in the meantime. Due to our experiences, this is not always the case. In the best case, the new VM is set up while the existing one is still running. However, the data has to be migrated from old to new, and depending on the amount of data this process could take some minutes. That is two VMs have to be paid during the transfer. A deeper empirical performance evaluation of candidates should be performed to clarify these issues. As an alternative, a larger machine with better equipment can be provisioned from the beginning, however, causing high upfront costs already for some few tenants.

If an upgrade could cause a downtime or other problems, one can apply a *scale-out* strategy which starts small with a single server and then extends the number of servers whenever the used servers get overloaded. Then, the expenses increase tenant by tenant. An incremental provisioning can help to avoid an upgrade and its consequences and also to reduce tenant-independent upfront investments, which occur already with the first tenant. Hence, the difference between n small DB servers and a less number of larger DB servers, having similar equipment, will be analyzed.

A. 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.

TABLE VII. PRICE SCHEME FOR OFFERING 5

Tier	*vCores*	*RAM*	*TempDisk*	*VM/month*	*DBS*	*#disks*
A0	1	768 MB	20 GB	$15	$298	1
A1	1	1.75 GB	70 GB	$67	$298	2
A2	2	3.5 GB	135 GB	$134	$298	4
A3	4	7 GB	285 GB	$268	$298	8
A4	8	14 GB	605 GB	$536	$595	16
A5	2	14 GB	135 GB	$246	$298	4
A6	4	28 GB	285 GB	$492	$298	8
A7	8	56 GB	605 GB	$983	$595	16
A8	8	56 GB	382 GB	$1,823	$595	16
A9	16	112 GB	382 GB	$3,646	$1190	16
D1	1	3.5 GB	50 GB	$127	$298	1
D2	2	7 GB	100 GB	$254	$298	2
D3	4	14 GB	200 GB	$509	$298	4
D4	8	28 GB	400 GB	$1,018	$595	8
D11	2	14 GB	100 GB	$600	$298	2
D12	4	28 GB	200 GB	$1,080	$298	4
D13	8	56 GB	400 GB	$1,943	$595	8
D14	16	112 GB	800 GB	$2,611	$1190	15

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. 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 database server (i.e., the software) is charged per minute (cf. column DBS in Table VII). The prices depend on the number of cores of the used VM: $298 for 1- to 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 are 5ct/GB-month for the first 4000 TB of consumed storage in a page blob; the price decreases by 0.5ct per GB-month for more than 4000 TB. The gain by using one VM instead of several ones is thus extremely small. In general, the costs are dominated by other factors than disk space.

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. Table VIII presents a brief evaluation of an SQL Server Standard Edition compares an A9 tier with several smaller machines summing up to the same price of about $4836.

TABLE VIII. CONFIGURATIONS COMPARABLE TO A9 VM.

configuration	*# vCores*	*RAM*
15,4 * A0	15.4	11.6
13,2 * A1	13.2	23.2
11,2 * A2	22.4	39.2
8,5 * A3	34.2	59.8
4,2 * A4	34.2	59.8
8.9 * A5	17.8	124.5
6 * A6	24.5	171.4
3 * A7	24.5	171.5
2 * A8	16	111.9
1 * A9	16	112

The configurations 8.5*A3 and 4.2*A4 obviously provide the highest number of vCores, more than twice compared to the reference 1*A9, while 6*A6 and 3*A7 offer the highest amount of RAM, again much more than 1*A9. That is, 1*A9 is not the best choice. Using A0s or A1s offers the least equipment because of the high portion of $298 for the database software for each instance, e.g., here 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). In general, it looks reasonable to avoid high-class VMs such as A9 and to use several middle-class VMs depending on whether favoring vCores or RAM. Moreover, a larger VM can be used for other purposes as well if being idle.

Similar considerations can be made to achieve a certain amount of RAM or number of vCores. For example, Table IX shows several options to achieve 112 GB RAM with very different prices and numbers of vCores. Analogously, 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.

TABLE IX. 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
1*A9	16	$3,646
2*A8	16	$3,646
1*D14	16	$2,611
8*A5	16	$1,968
2*A7	16	$1,966

Next, we investigate another scenario: We assume that a new tenant arrives each day. A scale-out strategy starts with an A0 instance, since this has the lowest price with $313, and to add further A0 instances whenever performance degrades. Let us assume that such a situation occurs with every new tenant, i.e., every tenant requires a new A0 VM. We compare this configuration with two others that immediately start with 1*A9 and 6*A6 respectively. Figure 6 shows the daily cost for each variant. After 15 days, we obtain the following cumulated costs:

a) 15*A0: $1271
b) 6*A6: $948
c) 1*A9: $2418

The cumulated costs for A9 are clear: that is exactly the price for half a month. In case of A0 VMs, we start with 1 VM for the first day, two VMs for the second day etc. In case of A6 VMs, we can use the first instance for the first 4 days (1 VM is comparable to 4*A0 wrt. vCores), and so on. Obviously, A6 VMs are a good choice. Using 6*A6 incrementally is cheaper than A0 VMs with the beginning of the third day. Already starting with a high-end A9 VM is about twice as expensive even after 15 days.

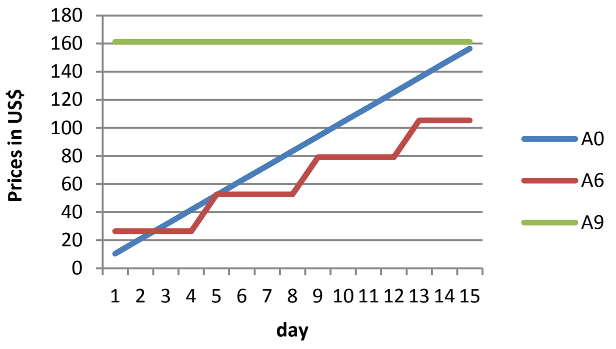

Figure 6. 15-day scale-out scenario for Offering 5.

B. *Offering 6*

Similar to Offering 5, this option again offers a VM running a database server, however, with some differences regarding pricing. Several database systems such as MySQL, Oracle, PostgreSQL, and SQL Server are supported in various database instance types.

TABLE X. PRICE SCHEME FOR OFFERING 6.

Category	*Instance type*	*Price / month*	*vCPU*	*RAM*
Standard	M	$478.15	1	3.75
	L	$594.95	2	7.5
	XL	$981.85	4	15
	XXL	$1960.05	8	30
Memory-optimized (MemOpt)	L	$744.60	2	15
	XL	$1489.20	4	30.5
	XXL	$2219.20	8	61
	4XL	$4409.20	16	122
	8XL	$8818.40	32	244

The prices depend on the chosen instance type, the type of database server, and the region. The prices for a SQL Server Standard Edition in the US East region are presented in Table X. The underlying VM and the SQL Server license are already included in the price. The instance type determines the number of virtual CPUs (vCPU) and the main memory. Please note that other database systems such as MySQL also support cheaper VMs in a micro edition.

An additional cost factor is the outgoing data transfer to the Internet: 1 GB-month is for free. The prices then start with 12ct/GB and decrease to 9ct/GB for data volumes exceeding 40TB. For instance, if we have 100 tenants with 512GB data transfer each (in total 50TB), using one database server for all tenants will be charged with $5836.80 (40 TB à 12ct/GB + 10 TB à 0ct/GB) while using 50 DB servers will end up with $6144 (100 * 512GB * 12ct/GB). Hence, the savings are $308, i.e., $3 for each tenant, for quite a high amount of data transfer.

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

(1) General purpose SSD for 11.5ct per GB-month with a range from 5 GB to 3 TB; 3 IOPS per GB are included (this is the base performance; 3,000 I/O requests per second are temporarily allowed).

(2) Provisioned IOPS for 12.5ct per GB-month and additional $0.10 per requested IOPS/month, with a range from 100 GB to 1 TB and 1,000 IOPS to 10,000 IOPS.

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

The upfront costs for each database server are determined by the minimal settings: The smallest installation in terms of cost for a SQL Server is Standard Medium (M) with $478.15/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 (1) above). Hence, setting up a minimal SQL Server, e.g., for each tenant, comes with at least $478.73 using general purpose storage, while provisioned storage is much more expensive with $590.65.

Again, we have to decide how many servers of what type are reasonable for a multi-tenant environment. We use a Standard XXL VM as a reference and compare it with several smaller VMs of the same overall price in Table XI.

TABLE XI. CONFIGURATIONS COMPARABLE TO XXL VM

configuration	*# vCores*	*RAM*
4.1*M	15.4	4.1
3.3*L	24.7	6.6
2*XL	29.9	7.9
1*XXL	30	8

The provided equipment differs a lot. Here, both 1*XXL and 2*XL offer the best price/equipment ratio.

Next, we again perform the 15-day scale-out scenario in Figure 7 comparing a scale-out with M, XL, and XXL for 5GB general purpose storage. We here consider 4*M being equivalent to 1*XL and 8*M equivalent to 1*XXL. We obtain the following cumulated costs after 15 days:

a) 8*M Standard: $1913
b) 2*XL Standard: $1178
c) 1* XXL Standard: $1437

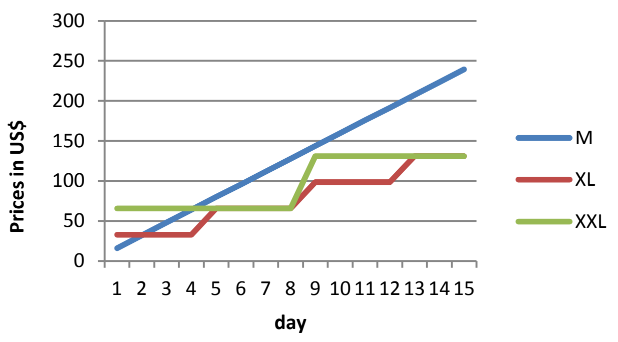

Figure 7. 15-day scale-out scenario for Offering 6.

Obviously, b) is the best option after 15 days, and starting with the third day it becomes cheaper than 8*M. XL benefits from an incremental provisioning compared to XXL. This also gives more flexibility for provisioning IOPS according to tenants' requirements.

A high-end server might be more appropriate since the network performance also increases with a higher instance class according to the documentation.

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.

Another 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 cost. Since there is an upper database limit of 1 TB in any case, general purpose IOPS ends with 9,000 IOPS; provisioned storage can handle up to 6,000 IOPS.

C. Offering 7

This offering provides a DB2 server in two editions: Standard and Advanced (Enterprise Server Edition). Standard corresponds to the DB2 Workgroup Server Edition, while Advanced is based on the DB2 Advanced Enterprise Server Edition, thus offering some advanced features such as compression, database partitioning, materialized query tables, query parallelism, multi-dimensional clustering etc. in addition. Both editions are offered for different tiers, S to XXL, each determining the physical hardware with the number of cores, RAM, IOPS, and attached storage. The S-L tiers use SAN, while XL and XXL both use SSDs and RAID10 for the first and RAID1 for the second disk array. Standard/Advanced S/M/L all have a 1 Gbps network, in contrast to a 10 Gbps redundant network for XL/XXL. Table XII presents the monthly prices for the region US.

TABLE XII. PRICE SCHEME FOR OFFERING 7.

Tier	*Cores*	*RAM [GB]*	*IOPS*	*DB Size*	*Price/month [US$]*	
					Standard	*Advanced*
S	2 à 2 GHz	8	100GB, 500 IOPS	100 GB + 500 GB	1,000	1,250
M	4 à 2 GHz	16	100 GB, 1200 IOPS	100 + 1000 GB	1,700	2,200
L	8 à 2 GHz	32	100 GB, 1600 IOPS	100 GB + 2000 GB	3,000	4,000
XL	12 (2.4 GHz)	128	10 Gbps	6 * 1.2 TB + 2 * 800 GB	6,000	8,000
XXL	32 (2.7 GHz)	1024	10 Gbps	16 * 1.2 TB + 2 * 800 GB	-	16,000

TABLE XIII. CONFIGURATIONS COMPARABLE TO XL VM.

configuration	*# vCores*	*RAM*	*IOPS*	*DB Size*
6*S	12	48	3000	3600
3.5*M	14	56	4235	3882
2*L	16	64	3200	4200
1*XL	12	128	-	8973

Again, the major concern is how many instances of what type should be used. The decision between Standard and Advanced must be based upon functional requirements and is not mainly affected by multi-tenancy cost aspects.

Table XIII compares several standard edition configurations the prices of which are the same as 1*XL. Again, the equipment differs a lot. 1*XL is best except for the number of cores, where 2*L has the lead. 2*L seems to be quite balanced. IOPS numbers are not given for XL.

TABLE XIV. CONFIGURATIONS COMPARABLE TO L VM.

Configuration	*cores*	*RAM [GB]*	*DB Size*	*IOPS*	*Price [US$]*
4 * S	8	32	2,400 GB	2,000	4,000
2* M	8	32	2,200 GB	2,400	3,400
1 * L	8	32	2,100 GB	1,600	3,000

Table XIV compares several variants for the standard edition with 8 cores and 32 GB RAM each. The alternatives are nearly equally equipped except the storage that is slightly decreasing. But concerning IOPS, 2*M seems to be the best choice. The L instance suffers here. However, one L machine is much cheaper than several smaller instances. Taking XL into account, the prices double L obtaining 4 times of RAM and about 4 times of disk space. The same behavior is visible for the advanced edition. XL has 2*12 virtual cores due to hyper-threading with 2 threads per core, and thus 3 times more virtual cores.

Every tier doubles the equipment of the previous one for less money more or less. As a conclusion, it seems to be reasonable to start with low upfront cost, i.e., an S instance, until performance, database size, or IOPSs reach its limit. Then an upgrade to the next tier becomes necessary.

In a 15-day scale-out scenario we compare a scale-out with S, L, and XL. Figure 8 displays the result. We consider 6*S being equivalent to 1*XL. After 15 days, we obtain the following cumulated costs:

a) 6*S: 4000
b) 2*L: 2200
c) 1*XL: 5400

Obviously, b) is the best option after 15 days, and starting with the third day it becomes cheaper than 8*M.

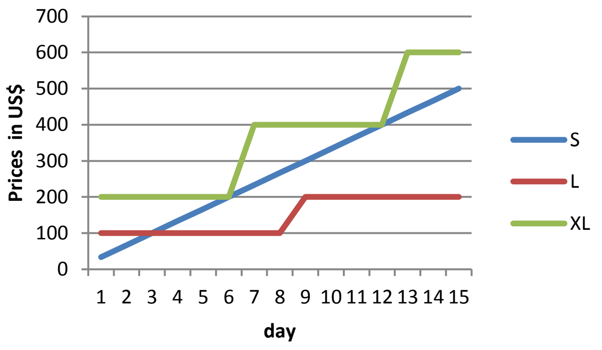

Figure 8. 15-day scale-out scenario for Offering 7.

D. *Offering 8*

This offering provides again a database server, in this case either Oracle 11gR2 or Oracle 12cR1. That is, a 1-DB-per-Tenant strategy is at no additional cost. Several choices have to be made:

- *Compute*: General purpose (GP) or High-memory (HM).
- *Editions*: Standard (SE), Enterprise (EE), High performance (HP), Extreme performance (XP). XP covers all the database enterprise management packs except RAC One Node, while HP has no Active Data Guard, in-memory feature, and RAC (Real Application Cluster).
- *Service level*: Virtual image or DBaaS (Database-as-a-Service). *Virtual Image* provides a compute environment with pre-installed VM images that include all software needed to run an Oracle server. All maintenance operations, including backup, patching and upgrades, are not automatically performed. *DBaaS* provides everything that the Virtual Image option offers, but also includes a pre-configured server through streamlined provisioning as well as automatic backup, patching and upgrades, and point-in-time recovery.
- *Virtual CPU* (vCPU): 1, 2, 4, 8, or 16. 1 vCPU corresponds to 1 physical Intel Xeon processor with hype-threading (or 2 virtual CPUs): 1 vCPU has 7.5 GB RAM for a GP instance and the RAM scales linearly up to 120 GB for 16 vCPU; a HM compute instance doubles the RAM compared to GP.

The choice of edition is only affected by functional requirements. Similarly, the decision for the service level depends on the desired level of comfort.

Table XV shows the monthly prices for an Oracle database server with its variants; it is important to note that these prices are per vCPU! Further costs occur for:

- compute: Each vCPU is charged with $75 per month (hourly metering);
- data transfer: 1 GB/month for free, then 12ct for each additional GB/month, increasing to 5ct for more than 150 TB;
- block storage: 5ct per GB/month;
- I/O requests: 5ct per 1,000,000 requests per month outbound.

TABLE XV. PRICE SCHEME FOR OFFERING 8 (FOR 1 VCPU).

Service level & compute / Edition	*DBaaS [US$]*		*Virtual Image [US$]*	
	GP	*HM*	*GP*	*HM*
SE	400	500	600	700
EE	1500	1600	3000	3100
HP	2000	2100	4000	4100
XP	3000	3100	5000	5100

For multi-tenancy considerations, the prices for data transfer, storage, and I/O requests can be neglected since they do not affect the per-tenant cost. There is gain for handing multiple tenants in one server in this respect.

Concerning the upfront cost, the smallest configuration is available for $475 (1 vCPU (SE, GP) for $75 + $400 for standard edition) plus storage, data transfer, and I/O requests.

One important observation is that a HM instance offers twice as much RAM as a GP, i.e., a standard edition with 1 vCPU HM compute is available for $500, while a standard edition with 2 vCPU GP is charged with $800, both offering the same 15 GB RAM. HM is thus a cheap option if RAM is important.

Quite obviously, the number of vCPUs is crucial for cost considerations, as the number is a multiplier for the cost of the DB server and the compute instance: 2 vCPUs double the prices for both and also RAM and cores. As a consequence, there is no cost difference between n * 1-vCPU machines and 1 * n-vCPU machine. Consequently, an incremental provisioning of 1-vCPU VMs for an increasing number of tenants is reasonable. That is, the 15-day scenario, comparing 1 vCPU wih 4 vCPU, yields some expected results presented in Figure 9. Using incrementally 1*vCPU is even on a per-day base starting with the first day cheaper, which let the cumulated cost become after 15 days:

a) 1 vCPU: $1900
b) 4 vCPU: $2280

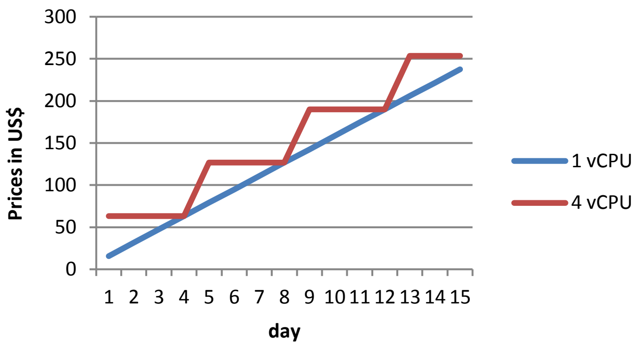

Figure 9. 15-day scale-out scenario for Offering 8.

However, one important question remains open: What is the advantage of a better n-vCPU instance, particularly compared to several smaller instances? Unfortunately, no specification beside the increase of RAM can be found in this respect. Further investigations are thus required.

VI. SUMMARY

Several differences have become apparent in the previous section and lead to several orthogonal facets:

- Database vs. database server: Sometimes, *databases* are paid (cf. Offerings 1-4); sometimes DB servers are provisioned (cf. Offering 5-8). The latter allows one to set up several databases within the database server, each database having specific credentials, i.e., tenant isolation comes for free. However, it becomes difficult to determine the right size of the VM, especially for avoiding upfront costs.
- Provision vs. consumption: Some offerings charge for *provisioned* storage (Offering 2-8), i.e., upfront costs occur even if small data is stored. Others (Offering 1) charge for storage *consumption* thus avoiding starting costs. However, the consumption approach is less prominent.
- Certain thresholds: Each offering has different tiers to choose from. Such a tier defines some thresholds. In most cases, each tier comes with an upper limit on the database size. Offerings 2 and 4 define certain limits on transaction rates, data transfer, or number of sessions, Offering 7 a limit on IOPS. Reaching such a limit could stop a SaaS application for serving tenants.
- For Offerings 3, 5, 6, 7, and 8, equipment such as RAM increases with each level, while this is not controllable and visible in Offerings 1, 2, and 4.
- Prices depend on the features such editions (Offering 7 and 8) or installation comfort (DBaaS in Offering 8). However, the decision is not affected by multi-tenancy considerations.

Moreover, we can distinguish between direct and indirect cost factors. *Direct* cost factors are immediately visible in the price scheme such as storage, IOPS, sessions, cores, or data transfer. 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, or IOPS (Offering 7).

VII. CONCLUSION NAD FUTURE WORK

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.

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 for virtual databases by 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.

Furthermore, we investigated the category of virtual database servers. Here, we could derive some vendor-specific strategies what category of database servers is suited.

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 [22] 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 [25].

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] U. Hohenstein and S. Appel, "The Impact of Public Cloud Price Schemes on Multi-Tenancy", in 7th Int. Conf. on Cloud Computing, Grids and Virtualization, Rome (2016), pp. 22-29.

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

[3] 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.

[4] 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.

[5] 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.

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

[7] 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

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

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

[24] P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, Sept. 2011. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf [retrieved: November 2016].

[25] 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.

[26] 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.

[27] 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.