Database Pricing: The Public Cloud and Its Impact

Hakan Jakobsson
hakan@hakan-jakobsson.com

Version 2015-08-01

Introduction

The impact of the public cloud will be huge for any of the established IT companies and not necessarily in a good way. They would include the likes of IBM, HP, Oracle, Dell, and Microsoft as well as many others. Of the old-guard players, Microsoft is probably furthest along in the public cloud game with Azure but hasn't been able to reestablish the "evil empire" status it used to have in the tech world in the 1990s. Instead the dominant player is Amazon Web Services, which is interesting given that Amazon didn't start out as an IT vendor. Other old-school vendors are busily jumping into the cloud game to prove that they are not dinosaurs with Oracle taking on AWS on pricing for tape-based cloud storage.
http://www.theregister.co.uk/2015/06/23/ellison_aws_price_war_amazon/
(Image wise, that may not be the best choice of battleground if you want to prove that you're not a dinosaur.)

Pricing is an important component in the database market, but it's hardly something that can be considered in isolation from issues such as features and functionality, support, third-party ecosystem, etc. And database software is extremely sticky; switching costs are high enough that a minor difference in price or performance – say 30 percent – is unlikely to persuade any sane customer to switch out a well-functioning production system. But a price difference of an order of magnitude or more is bound to have customers take notice, especially customers who are merely looking for mainly basic functionality. For all the headlines about companies with extreme data sizes or transaction rates, it's worth remembering that it's probably the outliers that are making the headlines and that a large fraction of the market tends to be more vanilla.

Another issue that is worth noting is one of transparency and openness. The old world of enterprise software sales was vey much one of proprietary software being sold by directs sales people and subject to all kinds of negotiations about discounts. That world is increasingly being challenged by open-source software and pricing transparency. In the cloud, AWS does a pretty good job of being transparent. And there are user communities eager to share tips about how to get the most mileage out of the different AWS pricing options (on-demand, spot price, reserved instances, etc.). That model is a universe away from having to negotiate your prices with the software equivalent of a used-car salesman. Now, AWS is expanding into the traditional enterprise software space with professional services, direct sales people

and negotiated deals (if they are large enough), but the official prices are probably pretty competitive and you can always look up the prices of competitors like Azure.

Still, as we will note, there are cloud service providers that are less transparent than AWS but likely at their own peril. A ZDNet article on the subject quotes several surveys showing that transparency is important to IT decision makers. "*Greater visibility is key if cloud is to move forward as a key technology resource. In the Forrester survey, 60 percent of IT leaders say lack of transparency has hindered their plans to expend cloud adoption.*" according to the article. http://www.zdnet.com/article/wanted-more-transparency-into-cloud-costs-and-usage-patterns/

Likely, competition and smaller deal sizes will gradually erode opaque pricing policies in the public cloud. Opaque pricing only makes sense if you want to get a salesperson involved but that may not be cost effective if the cloud service is sufficiently inexpensive.

Changing pricing dynamics

The dynamics of database pricing can change over time and that's hardly a new phenomenon. In the 1990s, Microsoft SQL Server emerged as a less expensive entrant in the database market and a growing threat to the established vendors as its scalability and functionality gradually improved. During the height of the dotcom boom, Oracle running on Solaris on top of large SMPs from Sun Microsystems was a popular combination among startup companies. After the dotcom bust, that combination fell out of favor as "free" open-source software gained in popularity, and startups might be running MySQL on Linux on less expensive hardware. The popularity of MySQL didn't go unnoticed among the large database vendors and in the 2005-2006 timeframe, Microsoft, Oracle, and IBM all introduced free versions of their database software with the word "Express" in the name. The threat from MySQL wasn't just about price but also convenience. Many companies have very bureaucratic procedures for software procurement, and a developer needing a database for a pilot project might find it easier to just download MySQL than going through the approval process to get an Oracle license even if the company was an Oracle shop. The Express editions were an attempt to prevent that from happening by providing a freely downloadable version of the database suitable for prototyping but artificially constrained so that it would be incapable of running larger production workloads.

Today, cloud computing has become the computational paradigm of choice for many startups as well as for many new projects within established companies. As cloud computing is increasingly gaining momentum, its competitive landscape is emerging complete with some rather disruptive pricing.

Cloud adoption: the carrot and the stick

As established IT vendors are aggressively trying to position themselves as having a major presence in the cloud, the question becomes how to actually establish such a presence. AWS has very significant lead over established IT vendors in cloud functionality and a gigantic lead in mindshare among Silicon Valley startups. So where are the established vendors going to get their cloud revenue? Acquiring cloud companies seems to be one popular approach, but it's expensive, cloud companies with high revenue are in limited supply, and acquisitions don't lend that much credibility to the cloudiness of a vendor's established, core technology.

So obviously, vendors are looking to take their existing on-premises install base to the cloud. This is where the carrot and stick problem comes into play. Established vendors may go heavy on the stick, both on their salespeople and customers, because using a carrot is tricky.

It has been claimed that Microsoft throws in Azure into deals for little or no extra cost even if the customer has no plans to use it. Just having an unused cloud component as part of a deal may be insufficient, however, in terms of credibility and could create accounting issues relating to revenue recognition. So supposedly, Microsoft is pressuring its sales force to make sure that customers actually consume their cloud credits by making cloud consumption goals part of the salespeople's compensation plans.
http://www.businessinsider.com/microsofts-cloud-consumption-problem-2015-3

Oracle, it has been claimed, uses audits of customers' use of its software to find breaches where a customer has used software beyond the licensing terms and is threatened with dramatic consequences unless it buys cloud licenses that it might not really want.
http://www.businessinsider.com/oracle-is-using-the-nuclear-option-to-sell-its-cloud-software-2015-7

Regardless of the veracity of the claims of a somewhat heavy-handed use of the stick to promote cloud adoption, it's pretty obvious why using a carrot is tricky for established vendors. IT organizations are extremely risk averse for good reasons. Switching an existing production system to a new computing platform is a high-risk proposition that also comes with significant cost. Even doing new projects in the cloud has cost related to establishing new procedures, acquiring new skill sets, etc. The most obvious candidate for a carrot would be if the risk and cost associated with going to the cloud would pay off in significantly lower cost going forward. In other words, a customer might be more likely to embrace the cloud if the result is that it's going to pay the IT vendor a whole lot less money. The IT vendor may not be all that enthusiastic about the prospect of reduced revenue and will be reluctant to make its cloud pricing sufficiently attractive until forced to do so by competition from pure cloud vendors that don't have cannibalization issues. So for now, it's

likely that the stick will be an important part of how established vendors promote cloud adoption.

Redshift as a baseline for cloud pricing

There are many types of database processing available as cloud services, but in what follows, we restrict ourselves to data warehousing as a service (DWaaS) and use Amazon Redshift as an example of the disruptive impact of the public cloud. Redshift is a managed relational database service for data warehousing. When it was rolled out in 2012, it came with the catchy pricing of $1,000 per TB per year for a managed service. (It applied to a certain instance type with a three-year commitment; other instance types were and are more expensive.) That price point made people take notice since it was generally considered that running a data warehouse would cost one to two orders of magnitude more if you did it in your own datacenter. And it made Redshift the fastest growing service in AWS history. It lacks quite a few of the features of the more mature databases, like Oracle, DB2, and Microsoft SQL Server, but the stuff it was designed to do – very fast basic SQL queries involving table scans against very large datasets – it does very well. It was originally created as a proprietary database, ParAccel, by modifying an open-source one (Postgres). In that sense, it may well fit in with current trends. In the years before the Redshift rollout, several upstart data warehousing companies like DATAllegro, Greenplum, Vertica, and Aster Data were acquired by bigger players like Microsoft, EMC, HP, and Teradata in deals typically in the $200-$400 million range. Amazon bought the right to the ParAccel source code for a tiny fraction of that. Redshift's sweet spot, scans of very large tables, may be only one of many possible sweet spots in the DWaaS market space, but one that fits well with the *Zeitgeist* of the era of "Big Data."

Let's look at the current Redshift pricing: If you use a Redshift cluster with ds2.8xlarge nodes, each node comes with 24 2-TB disks for a total of 48 TB of raw storage. It also comes with 36 cores and 244 GB of memory. Of the 48 TB of raw disk, AWS figures that one third will go to temporary space needed to perform large sorts or joins, to the operating system, etc. That leaves 32 TB for user data in the actual database tables, but because the data is mirrored, that space is reduced by a factor of two to 16 TB. So that's what AWS advertises as the capacity of a node. Cost per node with a three-year commitment: $44,840 if you pay upfront for the three years. That comes to $934 per TB per year.

What about data compression? Some vendors like to incorporate compression into their marketing pitch and make some assumptions about compression ratios when describing the capacity of a system. While those assumptions may often be believable and perhaps conservative, the fact that AWS doesn't do so in its $1,000-per-TB-per-year marketing pitch feels refreshing. And Redshift supports all kinds of data compression, but in AWS marketing, that's just icing on the cake.

At this point, it should be emphasized that the Redshift pricing is not based on the amount of data that is stored per se – something one might be led to believe by the

phrase "$1,000 per TB per year." Rather, the pricing is based on instance type and type of commitment. And each instance type comes with a certain number of cores, a certain amount of memory, and a certain amount of storage, so the per TB price is derived from the characteristics of the instance type. This point is important when comparing Redshift to products with very different pricing models.

Oracle the old way

Now let's consider Oracle, the market leader in database software using its traditional pricing for software licenses. In this case, we will notice that the per-processor pricing is based on the number of cores of the database server, not the amount of data in the database. So let's price out Oracle for the same AWS hardware on a per node basis using Oracle's price list using the kind of Redshift node that can hold 16 TB of user data at $16,000 per year.
http://www.oracle.com/us/corporate/pricing/technology-price-list-070617.pdf

You would probably want the Enterprise Edition of the database with the RAC and Partitioning options at the very minimum. RAC, since we are running on a cluster; Partitioning, since Redshift has range-restricted scans. Oracle has many other options as well, but let's stick to those. The per-processor license comes to $47,500 + $23,000 + $11,500 per Oracle's price list. Now multiply that by the number of cores, 36, and apply the core factor. The lowest core factor I have seen in Oracle's official listing, which lists it for different types of hardware, is 0.25. So let's go with that number and that brings us to $730,800 per node in upfront license fees just for the software. You will have to pay for the hardware as well.

But wait, Oracle also charges annual fees for "Software Update License & Support" that are 22 percent of the initial license fees. Doing the math for the annual fees, we get $160,776 per year. So for a node with 16 TB of user data, the mere maintenance fees would runt at over $10,000 per TB per year just for the database software.

Now it should be said that nobody pays Oracle the full list price. There are discounts and larger deals are usually carefully negotiated. However, Redshift is a managed service whereas if you get Oracle, you have to pay people to do management tasks that the AWS infrastructure will do for you for free. And that infrastructure has great economies of scale. And we still haven't included any hardware cost for Oracle. Even if you believe that Oracle is a better product with more bells and whistles than a Swiss Army knife, people are likely to ask themselves if it's worth the high cost.

So from a pure price perspective, things don't look too good for Oracle. Could Oracle match the Redshift price point? That would take a huge toll on Oracle's earnings, sales force compensation, and lead to all kinds of internal turmoil.

Database-as-a-service pricing

In order to compare Redshift to DWaaS offerings from other vendors, some background discussion about database architecture might be useful. Other vendors often have different pricing models with separate costing for processing (or "compute") and storage. To what extent such pricing makes sense is dependent on database architecture and factors that also impact important cloud concepts such as elasticity.

One service offering that doesn't separate processing and storage cost is Oracle's Exadata Service, part of Oracle's cloud initiative. Instead, pricing is for a quarter, half, or full rack each with a certain number of cores and a certain amount of memory and storage space. That model has some similarities to how Redshift is priced by instance type. So we will discuss the history and architecture behind Exadata as a background for examining pricing models.

Exadata background

Before Oracle rolled out its Exadata product in 2008, Oracle databases essentially consisted of servers connected to disk arrays where Oracle was just a piece of software on the servers. Oracle, with its history of portability, would support all kinds of hardware combinations and actual Oracle data warehouses were often great examples of that. An IT department might have different preferred vendors for different kinds of hardware, e.g., HP for servers, EMC for storage, Cisco for networking equipment, etc. Often, different people would be in charge of each area leading to potential political conflicts when deciding on the hardware components of the database. Dealing with sales people from different vendors with different agendas probably didn't make design decisions any easier. Quite often, the result was an unbalanced configuration where some component was undersized relative to the others leading to obvious performance bottlenecks. Such scenarios were one motivating factors behind Exadata since Oracle was competing against database appliance vendors, like Netezza, that were selling hardware and software bundled in supposedly completely balanced configurations.

Another consideration, perhaps more important for the purpose of this discussion, was the utilization of the disk array CPUs. Disk arrays would have CPUs that would mainly be responsible for storing or retrieving data on the disks as requested by the software on the servers. As CPUs were getting faster, it was natural to consider using them for something more intelligent than just processing I/O requests from the server. The storage CPUs had a much more direct connection to the data on the disks than the database server CPUs, so it was natural to consider using them for some form of database functionality. One type of functionality that was considered and implemented was the ability to filter out data that wasn't needed when processing a database query. A database table might have many columns, but a given query might only need a small subset of them. The same query might also contain conditions that only a subset of the rows in the table would satisfy. So rather than sending the entire table to the database server and have the server CPU figure

everything out, the storage CPU could figure out what part of the data was relevant and only send that subset to the server. That required the storage CPUs to run code that would understand Oracle's block format and could evaluate query conditions – code that had previously only been executed on the server. The benefit was a reduction in the workload on the server CPUs and a reduction in the amount of data that had to be sent over the interconnect to the server, and the result was a very large performance improvement.

However, pushing some of the database processing into the storage cells meant that processing and storage were no longer as separated as before; they were tied together in the interest of superior performance.

So how do the two processing models, Oracle Exadata and Oracle non-Exadata, relate to pricing models for cloud services? When processing and storage are bundled together physically, it makes more sense to charge for the chunk of hardware that performs both duties rather than trying to separate them out. Redshift, where the data is stored on local disks, bundles processing and storage physically and is priced per node. The pricing unit for Oracle's Exadata Service is a rack, which is also a processing and storage bundle.

Architectures with storage and processing in separate components would be more natural candidates for pricing models based on the usage of the individual components. One could, of course, measure the usage of Redshift CPUs that are tied to storage and charge for the CPU time used. But if the CPUs are so tightly tied to the storage that they couldn't be used for other, unrelated, purposes, what would the point be of charging for their usage separately from storage?

The discussion about separation of processing and storage also ties into the important cloud concept of elasticity, the ability to allocate more resources when you need them. From a pricing standpoint, the converse is just as important: If you can deallocate resources when you don't need them, you don't have to pay for them.

Storage elasticity is harder than processing elasticity if performance is a factor, as it tends to be. If the data is distributed between the disks for optimal performance, changing the number of disks may require redistributing the data, a process that can be time and resource consuming. Increasing or decreasing the processing power in the form of the number of CPUs is relatively easy if the compute nodes are separate from the storage nodes *and* the interconnect never becomes a bottleneck. Hence, Redshift's bundled architecture is arguably less elastic than architectures where the processing power can be changed without affecting storage, and competitors have been using that in their marketing. However, the separation of computation from storage is subject to both limitations and tradeoffs, something that we will discuss in the next section. And a very crude separation could still be done with Redshift: Archive your rarely used, historical data, on dirt-cheap AWS storage outside of Redshift. On the rare occasions you need that data, spin up some Redshift nodes and import the data for processing. That's hardly a fast and convenient process, but

likely inexpensive. While Redshift is $1,000 per TB per year for storage and processing, pure storage is about $360 per TB per year on S3 and $120 on Glacier.

Locality tradeoffs

In a perfect world, there would be no benefits to locality between CPUs and disks. Storage would just be the same black box for all processing units, and processing units would be allocated as needed – why pay for unused processing power for large amounts of historical data if the data mainly just sits there on disk and you very rarely query it? That's a large part of the selling point of separate processing and storage pricing but it only makes sense if the underlying architecture supports the separation so that the pricing separation is in line with the service provider's cost for the service.

Intuitively, it would stand to reason that doing processing in direct proximity to disk would have very substantial performance benefits and, historically, that has very much been the case. However, arguments have been made that things could be changing. Network speed is increasing at a faster rate than disk transfer times diminishing the difference between a local and a remote disk access; increased use of compression to handle large data volumes has an impact on the tradeoff between disk storage and CPU cycles for compression and uncompression and making disk-I/O relatively less important. So go some of the arguments.

An AWS blog post about new instance types
http://blogs.aws.amazon.com/bigdata/post/Tx3RD6EISZGHQ1C/The-Impact-of-Using-Latest-Generation-Instances-for-Your-Amazon-EMR-Job
compared the performance of the old instance type, m2.2xlarge, and the newer r3.xlarge for a certain task. The blog post suggested that: "*It seems to be a common assumption that you get better performance when data is local to the cluster vs being remote in Amazon S3.*" In the subsequent comparison between m2.2xlarge using local storage, m2.2xlarge using S3, and r3.xlarge using S3, S3 very much held its own, possibly debunking the "common assumption."

Well, not so fast. The old m2.2xlarge has a single 850-GB disk. So S3 had no problem competing with a single disk drive. But what about a cluster of 100 Redshift nodes with a total of 2,400 local disks? Could you just hook up your 100 nodes to S3 and hope to scan your tables just as fast? Probably not. Those 2,400 Redshift HDDs that would come with 100 nodes are a reflection of the fact that you have 100 nodes and that Redshift will stripe the data over all of them. (Well, except the leader node.) So you have 3,600 cores sucking data from 2,400 local disks as opposed to from S3. That may make a difference.

So part of the performance issue with elasticity is related to how the data is distributed. If Redshift's elasticity model had been to add a new node as needed and put all the new data on that node, elasticity would be easy but performance would suffer. Database queries primarily tend to involve recent data and in this elasticity

scenario where new data would involve a new node, other nodes wouldn't be able be able to contribute their processing muscle without expensive data reshuffles between nodes during query execution. Instead, Redshift distributes the data over all nodes so that queries against the newly loaded data can easily take advantage of the processing power of all nodes. That model has its own set of issues, but it does allow for superior single-query performance at the cost of making elasticity more cumbersome – when you add a node, the existing data needs to be redistributed.

The fundamental observation about locality, as in using local disks, is that it's about the scalability of performance. At the high end, compute-storage separated architectures may have difficulties competing, both in general, but, perhaps more particularly, on price. However, in the scenarios where compute-storage separation is feasible, there is no doubt that there are advantages when it comes to pricing and elasticity.

When it comes to pure scalability, locality is king for pure mathematical reasons. Let's say you have one node with local disks. As long as every disk access you need is local to every node, you can scale that single node to 1,000 nodes at essentially a linear cost. But what if you have a model of a black box of compute units communicating with a black box of storage units. The complexity of communicating between *x* compute nodes on the compute side and *y* storage nodes on the storage side becomes related to *x*y* if you want performance to stay exactly the same. Not a happy development as *x* and *y* grow large.

So database architectures, like shared-disk, where you typically separate processing and storage are great when you can get away with it. So the question is how big is the part of the market where you can get away with it and how big is the part of the market where you can't for scalability reasons. A historical comparison would be the debates in the 1990s, both in academia and in the commercial marketplace, between proponents of shared-disk and shared-nothing database architectures. As database sizes continued to grow larger and larger, some shared-nothing proponents declared that their architecture was the only way to go. Still, thanks to continuous hardware improvements and software innovations, Oracle with its shared-disk architecture did pretty well in keeping up with the increasing data volumes and managed to increase its market share. Today, both architectures are very much alive.

Lack of price information

One difficulty in comparing DWaaS pricing is that some vendors either don't publish a pricelist or fail to give the technical specifics of what you get for the price. That is surprising for two reasons.

1. It makes those vendors that are less than forthcoming with information look shifty compared to those that openly publish prices and technical specifications like AWS and Oracle; one can hope that competitive pressure will gradually force increased openness.

2. The reason for opaqueness is obviously to get salespeople involved, but a heavy involvement of salespeople may not be a sustainable model given the deal sizes. In traditional database sales, you would often have long sales cycles, time-consuming proofs of concepts, price negotiations, etc. handled by a direct sales force. But direct sales are costly and require relatively large deal sizes to be profitable. It's also harder to scale up your sales if it requires hiring and training more salespeople. That's why the preferred sales model for the smaller deals would be to use other types of sales channels. If in the public cloud, the DWaaS deal size for a 10 TB data warehouse is $10,000 per year in revenue, there is little room for paying a whole lot to sales people to get involved. AWS can achieve a large volume of such deals with minimal involvement of sales people, which makes you wonder how well those vendors that cannot do so will be able to compete.

Exadata Service pricing

Having discussed Exadata extensively, we can now look at how Oracle prices it as a service and compare it to Redshift.

Oracle's smallest unit of Exadata Service is a Quarter Rack, which can have a minimum of 28 "OCPUs" and a maximum of 68 priced at $5,000 per OCPU per month. So for the minimum configuration, that's $140,000 per month or $1,680,000 per year.
https://cloud.oracle.com/database?tabID=1406491812773

A Quarter Rack comes with 42 TB of "usable storage." It's not clear whether some of that would have to be used for things like temp space, but let's say it's all for the user's table data. That would come to $40,000 per TB per year – quite a bit more than Redshift.

It should also be noted that Oracle's Exadata Service isn't a managed service in the same sense as Redshift. However, Oracle's website has managed service as a "future direction" that will include Oracle-managed backups, patching, and upgrades, functionality that is already part of Redshift's managed service.

Teradata pricing

Teradata doesn't need much introduction. It's one of the major players in data warehousing with a hardware-software combination product that is considered solid albeit very pricey. It also offers data warehousing as a cloud service with published pricing. It's pricing is per Cloud Compute Unit (CCU), which comes with a certain amount of both processing power and storage just like the Exadata's racks and Redshift's cluster nodes.

Teradata doesn't tell us exactly what a CCU is (It wouldn't be like them to volunteer such useful information.), but we learn from the FAQ that it comes with 4TB of

usable storage (before compression) and that Teradata recommends using at least two for failover protection. So it sounds a lot like nodes in a cluster, which is exactly what you would expect based on Teradata's architecture. Pricing: Teradata has a pricing option based on a three-year commitment just like Redshift. Per CCU, it's $24,000 in reservation fee and $3,700 per month. Over three years, that's a total of $157,200 or $13,100 per TB per year. That's 14 times more than Redshift's current $934 per TB per year although I'm sure Teradata will tell you that it's worth every penny.
http://www.teradata.com/Layouts/IndustryLayout.aspx?pageid=12884910014

One would wonder, however, how genuinely interested Teradata is in actually selling its cloud service as opposed to having one just to reassure its customers that the company is on top of recent technology trends. The trailblazing adopters of the public cloud are often into openness and transparency. Teradata seems to extend its old-school IT sales tactics – if you want any useful information, you need to talk to a sales rep – to its cloud service. And even at $13,000 per TB per year, it may still be preferable for Teradata to have a sales rep try to nudge a potential customer to go with an on-premises solution. Very likely, Teradata itself doesn't have any definite answers as to how to best deal with the disruption that the public cloud is causing.

Google Cloud BigQuery pricing
Google's BigQuery is the external implementation of Dremel as a cloud service. The pricing is interesting in that it separates storage from processing, but not by charging for processing CPU power. Instead, the processing part is based on how much of the base data is needed to resolve a query. The storage part is simple, $0.02 per GB per month or $20 for storing 1 TB for a month. Not too bad.
https://cloud.google.com/bigquery/pricing

The query part makes things more interesting, $5 per scanned TB. That means that it only takes 4 queries scanning the 1 TB that you just uploaded to cost you as much as storing it for a month. Many production data warehouses have many hundreds of canned reports that are routinely executed as queries against the newly loaded data every night. Most will probably not need all the columns of every table containing the new data, but they might access previously loaded data in the form of reference data about products, customers, etc., as well as historical data for time-period-over-time-period comparisons. All of that makes it hard to predict the cost of Google's offering. If you want to use it as a write-only database, it seems pretty reasonable. Less so in so far you want to do serious query processing.

Another aspect is the whole quirkiness surrounding the idea of charging queries by the underlying data needed. To quote from the pricing URL above: "*When you run a query, you're charged according to the total data processed in the columns you select, even if you set an explicit LIMIT on the results.*" So if you join two data sets of 1 TB each in the columns you need, the size of the result could theoretically be 1 TB squared or 1 trillion TB (if you could join the data sets on a per-byte basis). So if Google were able to perform such a query (unlikely) and were to charge you for the

size of the output (either sent back to the users or used as the input of another database operation like a another join or a GROUP BY), you would have used up $5 trillion of your startup's Series A funding on just one database query. Fortunately, Google uses the input sizes of a query rather than the output sizes for its costing, but this somewhat unrealistic example shows that the input sizes of a query are not always a good measure of the actual processing cost that has to be paid for by someone.

Another issue is how well the pricing matches up with customer expectations. Two example queries from Google's price list are

```
SELECT corpus, word FROM publicdata:samples.shakespeare
LIMIT 1
```

and

```
SELECT COUNT(*) FROM publicdata:samples.shakespeare WHERE
corpus = 'hamlet'
```

Google tells us that the cost of either query would be based on the total size of the underlying columns even though there are plenty of database query processing techniques that could theoretically have resolved these two queries without necessarily scanning the entire underlying dataset.

So the Google query pricing has three obvious issues:

1. Does it strike the right balance between storage and processing to be attractive?
2. Does anyone who is not a SQL guru understand it?
3. Does it accurately reflect the cost to Google of executing the query?

The second question is important because charging based on scanned data sizes takes pricing based on usage in an interesting direction and one that a lot of people might have a hard time understanding. If you want to charge for processing, compute time might be a more intuitive concept. If you charge by time, a customer will know in advance how much it will cost to run a workload for three hours. Cost based on the amount of data scanned seems harder to predict. It's very possible that the Google pricing model is too convoluted for its own good.

The third question is important since, most of the time, a business would probably want to charge its customers based on the underlying cost plus a markup. So how well does the price Google charges you for a query track the actual cost to Google? The first question one would ask is this: If a user is querying the data interactively and there is a lull between queries, can Google use the hardware resources for other purposes or will they just be idle? If they just sit idle and without the user paying for any scan cost, it's a bad fit for the cost to Google. Other than that, my take is that the cost model is a very rough approximation of the cost to Google since it doesn't take into account the work needed for operations like joins and GROUP BY aggregation or for functions used in queries. If, at some point, the SQL supported were to get

extended to allow user-defined functions in queries, arbitrarily compute-intensive functions could be evaluated for every single data item scanned so the current model wouldn't be a great fit for that. So the query-processing cost model is probably less exact and more confusing than simply charging for compute resources based on the time they are allocated.

Microsoft Azure SQL Data Warehouse

Microsoft uses separate storage and processing when it comes to pricing and elasticity as a major selling point. Clearly, Redshift is the target.
https://awsinsider.net/articles/2015/04/30/microsoft-data-warehouse.aspx

So what is the pricing? According to the Azure website, it's $521 per month per 100 DWUs for processing while the service is in preview mode.

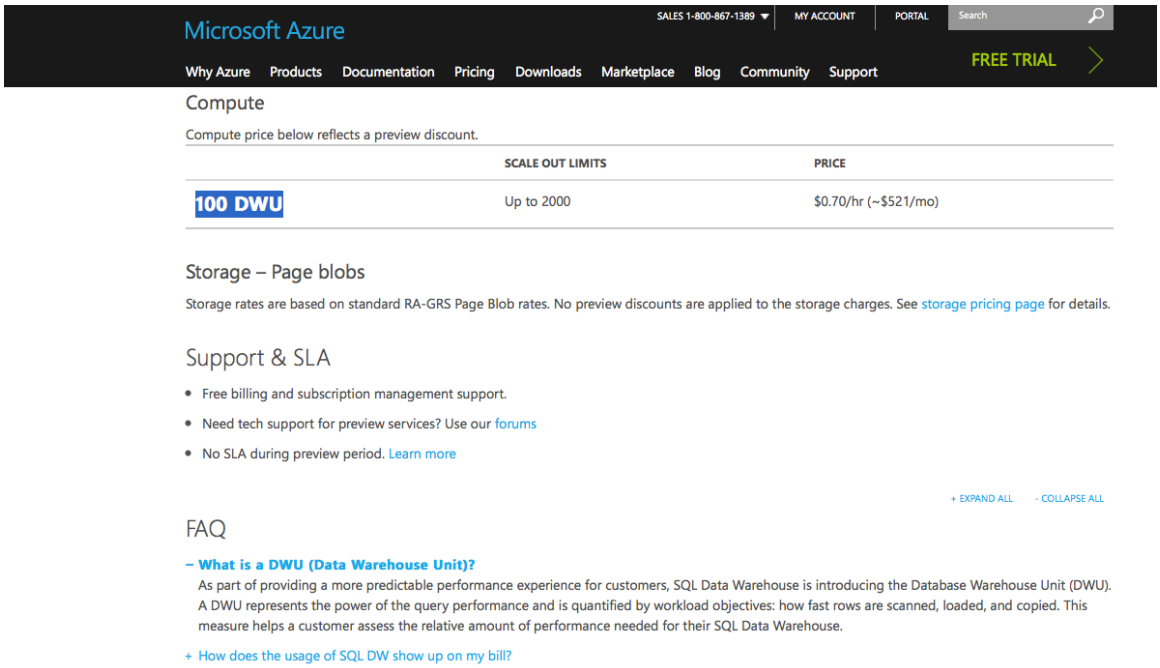
So what exactly is a DWU of which you get 100 per month for $521? According to the FAQ:

-*What is a DWU (Data Warehouse Unit)?*

> *As part of providing a more predictable performance experience for customers, SQL Data Warehouse is introducing the Database Warehouse Unit (DWU). A DWU represents the power of the query performance and is quantified by workload objectives: how fast rows are scanned, loaded, and copied. This measure helps a customer assess the relative amount of performance needed for their SQL Data Warehouse.*

Yep, that's what a DWU is. In case you thought of Teradata as being the king of the less-than-forthcoming vendors and Google as being confusing, think again!

From http://azure.microsoft.com/en-us/pricing/details/sql-data-warehouse/



The storage-pricing link is to the standard Azure storage page where pricing comes to a few hundred dollars per TB per year depending on the type of redundancy.

Using AWS as your hardware

In theory, you can run your own database on top of AWS and that raises the question of what the cost of the hardware rent is for running a database on EC2. There are DWaaS vendors who use AWS as their hardware platform so one could ask if you want to run on EC2 on Redshift-like hardware, how much would you have to pay Amazon?

Redshift's below $1000 per TB per year price is based on the instance type ds2.8xlarge with a three-year commitment that costs $44,840 for the three years if paid upfront. EC2 has an instance type called d2.8xlarge that is the equivalent of ds2.8xlarge in terms of cores, memory, and disks. Three-year upfront cost: $41,650. That's 93 percent of Redshift. The on-demand prices come in at $6.80/h and $5.52/h for an 81 percent ratio. The smaller HDD-based instance ds2.xlarge and its EC2 cousin d2.xlarge have similar ratios. The two SDD-based instance types don't seem to have any exact EC2 equivalents but we could compare dc1.8xlarge (32 cores, 244GB memory, 2.56TB user data SSD space) and the EC2 instance i2.8xlarge (32 cores, 244 GB memory, 6.4TB raw SSD space). In this case, the EC2 instance comes in at 12-42 percent more expensive than Redshift depending on pricing type.

So just renting the hardware might cost you at least 80 percent of the cost of getting Redshift as a managed service. And Redshift will do automatic backups to S3 without charging you for the S3 space. With your own database, you probably would

have to pay for the S3 space and pay for someone doing the backups for you as well as performing other tasks that will be included in the Redshift price as a managed service. So if all you are doing is something that is a good fit for Redshift, there would be very little upside in running your own database on top of AWS hardware. Vendors that provide DWaaS had better come up with significant differentiators that would allow for charging a premium compared to Redshift since Redshift's pricing won't lend itself to huge margins.

AWS has significant expertise in building infrastructure for automating the management of its gigantic infrastructure and can get tremendous economies of scale for the managed-service aspect of Redshift as well as from its gigantic hardware infrastructure – the economies of scale that allow Amazon to sell Redshift at low price and still make a profit. Redshift builds on the AWS infrastructure for provisioning, billing, etc. with some Redshift-specific extensions related to database operations like backup and recovery. Vendors who want to offer their own DWaaS product running on AWS must duplicate some of the "as-a-service" infrastructure that is already in place for Redshift that lets AWS manage its customers and the customers manage their databases. In addition, the database technology itself had better be either better than or different from Redshift and any new database offering AWS might be working on in secret. The possibility of AWS working on new services should not be forgotten if you want to avoid getting blindsided by it.

Let's do some math. Let's say that your company is providing DWaaS on top of AWS and uses a Redshift-like pricing of $1,000 per TB per year. The company manages to attract 1,000 customers, each with a 100-TB data warehouse. Most database companies that started in the last 20 years haven't been nearly that successful. Netezza, which managed to go public, had 373 customers in 2010, the year it was acquired by IBM. But let's say 1,000 customers. That would give you $100 million in annual revenue. If Amazon siphons off 80 percent of that in hardware rent, you are left with $20 million per year, not a whole lot given the success of the company. In traditional software sales, once you had paid off the cost of development, each additional license sale was very profitable if you used inexpensive sales channels. In the world of DWaaS, it will be less so if you have to pay an Amazon tax at a marginal rate of 80 percent on every additional revenue dollar.

In any case, competing with AWS with services similar to what AWS is offering doesn't seem like a business model with very large margins. Differentiation would be the key and it seems like compute-storage separation for pricing and elasticity and support for semi-structured data and external data sources are the most popular differentiation candidates right now.

BitYota

Founded in 2011, BitYota rolled out its DWaaS offering at the AWS re:Invent conference in 2012 apparently unaware that AWS would introduce Redshift at the very same event. The underlying database technology is apparently based on Postgres and runs as a service on top of both AWS and Azure. BitYota's marketing

emphasizes the separation of storage and processing without specifying the cost for either. Initial press reports mentioned a starter service with 10 ECUs of computing power and 500 GB of storage for $1,500 per month.
http://www.theregister.co.uk/2012/11/28/bityota_saas_data_warehouse_service/

This article from 2014 claims 20 employees and $12 million in funding.
http://www.enterpriseappstoday.com/data-management/startup-spotlight-bityotas-data-warehouse-as-a-service.html

Snowflake

Snowflake is another DWaaS company that runs its offering on top of AWS. Separate storage and processing elasticity is emphasized as a selling point. Unlike BitYota, the founders were aware Amazon was developing Redshift prior to founding the company in 2012. There doesn't seem to be any published pricing. As of June, 2015, it has received $71 million in funding and has 75 employees.
http://fortune.com/2015/06/23/big-data-startup-snowflake/

According to its CEO: "*So far, 80 customers have signed up to use Snowflake, and as many as 20 have moved beyond the testing phase.*"
http://recode.net/2015/06/23/big-data-startup-snowflake-raises-45-million-launches-first-product/

Conclusion

The AWS Redshift data-warehouse-as-a-service offering is likely an example of how public cloud computing will be highly disruptive over time, although of all its impact won't be felt overnight. The enormous economies of scale enjoyed by AWS combined by the fact that it's not cannibalizing any of its on-premises software products (since there aren't any) will likely disrupt the likes of Oracle, Microsoft, IBM, and Teradata over the next 5-10 years. Database software tends to be extremely sticky and customers will be reluctant to swap out existing systems, so the disruption will mainly be in setting price levels for new deals. Vendors that want to offer data warehousing as a service had better figure out how to differentiate their products from Redshift if they want good margins.

There are also implications for business models. Traditionally, software has had very high margins when sold in large enough quantities since the cost of creating additional copies of software in negligible. Selling the same software as a service requires providing additional hardware resources for each new customer, something that has an impact on how margins work out. Moreover, the deal sizes involved and sales structure as subscriptions rather than license sales may impact the roles and involvement of sales people. One can also note that if AWS charges customers $10,000 per year to run a 10 TB data warehouse, that cost will likely be a small fraction of the cost the customer will have for the people that design, operate, and use the data warehouse, not to mention the cost for other BI-related software that might be involved.