ORACLE COST TRAPS AND HOW TO OVERCOME THEM WITH AWS SOLUTIONS

Principal Author: Nick Walter, Principal Architect; House of Brick Technologies

Joe Grant, Principal Architect; House of Brick Technologies

Cam Cameron, Principal Architect; House of Brick Technologies

House of Brick Technologies, LLC received partial financial support from Amazon Web Services for the research and writing of this guide. However, the opinions and conclusions expressed in this paper are those of House of Brick, and not of any other company or individual. The third-party products referenced in this document, including those from VMware, Oracle, and AWS are copyrights of their respective owners. House of Brick consultants are not attorneys. All findings and recommendations, especially regarding Oracle licenses should be validated with legal advisors.

House of Brick
OPSCOMPASS GLOBAL SERVICES

TABLE OF CONTENTS

EXECUTIVE SUMMARY

Many organizations that want the benefits of a migration to AWS are hesitant about migrating Oracle database workloads. A variety of cost traps and risks need to be carefully assessed and managed to avoid exacerbating already high Oracle costs during a migration to AWS. The main Oracle database cost traps to consider are:

- The Oracle Licensing Cost Trap – Oracle licenses are costly and encumbered by complex licensing rules.
- The Oracle Change Fear Factor – Oracle database solutions are notoriously complex and fragile. Any upgrade or change that affects an Oracle database can be considered risky and act as a drag on organizational velocity and agility.
- The Oracle Database Operational Cost Trap – Oracle databases are complex and high touch. They require dedicated groups of specialized personnel to operate at scale that leads to significant overhead.
- The Organizational Silo Trap – The requirement for specialized teams and tools can create a silo that diminishes organizational velocity and agility.
- The Organizational Overhead Trap – Managing the complexities of Oracle licensing requires significant overhead, especially during a compliance audit.

House of Brick, drawing on extensive experience of helping clients migrate Oracle database solutions to AWS, has proven techniques for addressing all these risks. Utilizing these techniques, it is possible to plan a cost-effective migration to AWS for Oracle databases. These techniques include:

- Leveraging Amazon RDS for Oracle – Utilizing a managed database service like Amazon RDS for Oracle can simplify ongoing operations and directly address the Oracle Change Fear Factor, the Oracle Database Operational Cost Trap, and the Organizational Silo Trap.
- Migrating to open-source databases – Migration to an open-source managed service such as Amazon Aurora comes with some cost. But the potential savings in Oracle licensing can more than justify it. This is a powerful technique for avoiding the Oracle Licensing Cost Trap and the Organizational Overhead Trap.
- Rightsizing Oracle databases in AWS – Overprovisioning, an ingrained habit for most database teams, does not make sense in an AWS environment where capacity can be adjusted rapidly to meet demand. Utilizing appropriate rightsizing techniques goes a long way towards avoiding the Oracle Licensing Cost Trap and the Organizational Overhead Trap.
- Effective strategies for Oracle license wind down – Oracle licensing rules are complex and it is easy to inadvertently reduce Oracle usage without reducing costs. Using House of Brick's method to plan a license wind down maximizes savings and addresses the Oracle Licensing Cost Trap.
- Adopting DevOps techniques for Oracle databases – Utilizing modern DevOps techniques and rich automation can improve efficiency of Oracle operations and reduce overhead costs. This increase in efficiency is an effective way to avoid the Oracle Database Operational Cost Trap, the Oracle Change Fear Factor, and the Organizational Silo Trap.

AWS migration offers significant cost savings for entire IT organizations, so it is important not to let the high costs and complexity of Oracle licensing derail these opportunities.

House of Brick strongly recommends that any organization considering a move of application stacks built around Oracle database technologies to AWS consider and apply the principles outlined in this paper.

INTRODUCTION

Oracle database products are a costly part of the IT landscape for many organizations. Oracle databases, while feature rich, are complex and technically difficult to operate, which decreases the velocity of change and adds risk to projects that rely on them. Even worse, enterprise RDBMS vendors such as Oracle charge steep prices for software licenses and support while counting on vendor lock-in and a reputation for aggressive audits to keep customers paying expensive annual fees. Using Oracle database products comes with financial risks.

Oracle databases have a reputation for being complex and fragile. They require specialized personnel and dedicated team, which leads to the formation of silos that impair organizational agility. This frequently results in Oracle database related projects encountering unexpected delays or budget overruns.

Considering these risks, it may seem daunting to migrate Oracle databases to AWS. Embracing the advantages of AWS is extremely attractive from both a technical and cost perspective, but many organizations are hesitant when it comes to moving Oracle databases. Migrating mission critical Oracle databases to AWS can improve cost-efficiency, performance at scale, and operational velocity. Armed with the right knowledge and expertise, House of Brick recommends any organization looking to move their Oracle databases to the public cloud should consider AWS as their first choice.

House of Brick has helped many organizations rein in their Oracle costs with AWS services such as Amazon Relational Database Service (RDS) for Oracle for fully managed Oracle databases, or Amazon Aurora for fully managed open-source compatible database service that requires no database licensing fees, or even Amazon EC2 for customer-managed deployment of Oracle databases. In this paper House of Brick draws on its experience of advising and assisting clients worldwide with their migration of Oracle workloads to AWS to help identify all the cost traps and risks around Oracle licensing and technologies, and implement best-practices, solutions and strategies to minimize them in AWS.

ORACLE COSTS AND RISKS

The true costs of running an Oracle database workload are multifaceted. Even the obvious costs such as the purchase price of the license and the ongoing costs for support tend to have lock-in traps that are difficult to escape. Before charting a course to reduce or eliminate Oracle costs in AWS it is important to understand their exact nature and sources.

Oracle Traps and Risks: Financial Costs

House of Brick recently had the opportunity to work with a major U.S. insurance company that faced serious Oracle challenges when moving business-critical customer facing applications to the cloud. They had completed all their due diligence, hired a staff of cloud specialists, and were ready to charge forward into the AWS cloud.

However, their first major challenge arose when they realized that fitting their Oracle database footprint into the cloud would pose significant licensing costs. The high-cost, per-core licenses they employed for their dense on-premises Oracle Real Application Clusters infrastructure did not translate cleanly to the cloud, especially as they tried to replace complex Oracle clusters with the Amazon RDS for Oracle service to separate out all databases, remove interdependencies, and streamline operations. At the last minute, they contacted Oracle to ask about extra licensing and discovered the additional costs would put the entire project more than $1 million over budget and potentially derail the entire project. With the project suddenly in doubt, the company reached out to House of Brick for help.

House of Brick was able to find ways to optimize their licensing, as this paper will detail later. But it is important to highlight this scenario as it illustrates a situation that many organizations encounter. Unexpected Oracle costs can derail even high-momentum cloud migration projects, which is why understanding them and knowing the strategies to mitigate them is so important.

Oracle Licensing Cost Trap

The most obvious cost that will spring readily to mind for anyone who has negotiated the purchase of Oracle database software licenses is the purchase price. Oracle Database Enterprise Edition software licenses start at a list price of $47,500 per processor, as of the time of this writing. A great deal of uncertainty and fear exists around the true costs of Oracle software licensing. House of Brick has heard many clients relate stories of Oracle representatives delivering ominous warnings about how licensing costs can double in AWS. These claims are dubious and, with a clear understanding of the fundamentals of Oracle licensing, it becomes apparent that these claims seem unfounded.

Oracle Licensing Fundamentals

Oracle database software licenses are sold by differing metrics with the most common being the Processor metric or the Named User Plus metric, which is based on a minimum number of physical processors. Many organizations do not have user counts that exceed the required minimums, so these minimums often define the number of licenses required.

The Oracle definition of processor from a software licensing perspective is misleading as the number of licenses does not correspond exactly with the number of processors in a physical server. It is instead an abstract metric calculated by summing all the physical processor cores for Oracle Database Enterprise Edition or sockets for Oracle Database Standard Edition databases in a physical server and multiplying that number by a core factor particular to the specific processor architecture of the server. The core factor is

currently 0.5 for x86 architecture Intel/AMD processors but is subject to change in the future at Oracle's discretion.

It should be noted that slightly different rules apply when applying Oracle processor license to cloud environments. Oracle has granted the ability, via its extra-contractual Licensing Oracle Software in the Cloud Computing Environment[1] document, to license an EC2 or RDS instance in AWS for Oracle products using a virtual CPU (vCPU) metric instead of the traditional server processor basis. The document states:

> Amazon EC2 and RDS **– count two vCPUs as equivalent to one Oracle Processor license if hyper-threading is enabled, and one vCPU as equivalent to one Oracle Processor license if hyper-threading is not enabled.**

When licensing by vCPU instead of physical processor, the rules about Processor Core Factors do not apply. Total vCPUs in a cloud instance are used to calculate the needed licenses as opposed to summing processor cores or sockets. Any organization that wants to take advantage of this extra-contractual grant of vCPU licensing rules on Oracle's part should do so with confidence as House of Brick has seen Oracle respect the policy during multiple license compliance audits.

This policy is not, however, the only way to license Oracle software products in AWS. Whenever the physical processors underlying an Oracle workload can be accurately assessed, such as when using dedicated hosts or the VMware Cloud on AWS, then licensing on the contractually defined physical processor basis is the preferred option. This is a useful distinction as certain products such as Oracle RAC are omitted from the Licensing Oracle Software in the Cloud Computing Environment document and thus must be licensed via traditional means.

In addition to the high acquisition costs for software licenses, the annual cost for Software Update License & Support (SULS) is considerable as well. The standard price for the annual SULS renewal is 22% of the original purchase price annually. That annual renewal payment also has a built-in year-on-year compounding uplift. Typically, this is a 4% annual increase. Only three times in the last 20 years has the annual U.S. inflation rate exceeded 3%. Thus, Oracle renewal costs grow over time in real terms, even adjusting for inflation.

With the compounding nature of the annual support uplift baked into the contracts, Oracle has built a system where it can get clients to gradually accept increasing payments to Oracle for support renewals. Most organizations scrutinize payments for renewal of support far less than initial purchases. Because some enterprise application stacks can have extended lifespans, House of Brick has seen the support costs of decades-old Oracle processor licenses spiral to the point where it would be financially advantageous to simply terminate the license and purchase a new one.

Oracle offers enterprise accounts an Unlimited License Amendment (ULA) as an alternative to tracking licenses on a processor basis. Software products purchased under a ULA can be used on an unlimited basis for a fixed period, commonly three years. At the end of the ULA agreement a process called certification happens where the actual in-use Oracle software is inventoried, and Oracle issues processor licenses equal to the certification inventory. This allows an organization with a ULA to continue using Oracle products from the time of certification but requires them to purchase additional licenses going forward if they want to grow their Oracle platforms. While ULAs initially sound like a good idea, their high prices and unfavorable terms make them less attractive. House of Brick rarely encounters a client with a ULA that would not have been better served by simply procuring processor licenses to meet their needs.

Beyond the initial purchase price and the Software Update License & Support (SULS) payments, there are other costs and traps lurking in Oracle contracts. Oracle contracts can be complex and difficult to review and understand. Numerous items in the terms and conditions can burden the flexibility of acquired Oracle

[1]https://www.oracle.com/assets/cloud-licensing-070579.pdf

software licenses. Some of these items include: License Sets, Pricing Following Reduction of Licenses or Support Level, additional database features (options and packs), and updated contract terms.

License Sets

One of the lesser-known rules of Oracle licensing involves the concept of a license set. This rule states that a client cannot terminate support on a particular Oracle product unless they are terminating support on all owned licensed for that product. This is an understandable move on Oracle's part as they do not track individual license application to individual databases on an ongoing basis. Typically, Oracle license entitlement is purchased on a perpetual license basis, meaning a de-supported license is still valid for use but cannot be further upgraded or supported by Oracle.

Oracle support would have no way of determining if a customer contacting them for support was reporting an issue with a supported product or a de-supported product. With license sets, it is possible to terminate an entire license for a product at any time, but not to terminate support unless all other owned product licenses for that particular product have support terminated at the same time.

This means that an organization that wishes to discontinue support for a portion of the total entitlement portfolio must cancel those licenses outright or alternatively cancel support for all licenses owned for that product. For example, consider an organization that owns 100 processor licenses of Oracle Database Enterprise Edition and wishes to cancel support for 20 of them. Due to license set restrictions, they must cancel support for all 100 or terminate the 20 licenses outright and completely cease using them.

Pricing Following Reduction of Licenses or Support Level

The Pricing Following Reduction of Licenses or Support Level clause in the Software Technical Support Policies document[2] referenced by the Oracle license agreement appears to be designed as a lock-in measure to make it difficult for an organization to reduce their Oracle spend. Nearly all Oracle license orders contain a discount from the published list price. This particular clause states that if support on any part of an order is cancelled, Oracle can reprice the continuing support payments for the surviving elements of the original order without the original discounts.

There is a cap in place that states repricing cannot cause the total support payment to increase, and support cannot rise above "Oracle's list price for support in effect at the time of termination or reduction minus the applicable standard discount." The net effect is to make it difficult for organizations to realize any savings on the expensive ongoing support costs by incremental reduction of their Oracle license portfolio.

To illustrate this clause in practice, assume an initial order of Oracle software licenses contains 10 processor licenses for Oracle Database Enterprise Edition at a 40% discount from list price, or $28,500 per license. Annual SULS payments of 22% of the purchase price after discounts are applied so that each license costs an additional $6,270 for the first year of support.

If an organization determines later that it only needed eight of the processor licenses and wishes to stop paying support on the excess two on the next SULS renewal, they face a dilemma. While it is quite possible to cancel support on those licenses, Oracle can reprice the surviving items from the original order to full list price with perhaps a small discount when calculating the next cycle's support renewal costs. As this re-evaluation will likely eliminate the 40% discount, it means that instead of paying $6,270 per license for 10 licenses, or $62,700, they may instead face a renewal price of $10,450 per license for eight licenses, or $83,600. While the repricing clause does not allow Oracle to increase the net SULS costs above the original

[2] www.oracle.com/us/support/library/057419.pdf

order, it is quite possible this cancellation will not achieve any actual savings unless the terminated licenses comprised more than 40% of the cost of the order.

Extra Options

From a technical perspective, the list of add-on features (options and/or packs) for Oracle Database is a welcome addition. Database Options or Management Packs such as Partitioning, Advanced Compression, or Advanced Security allow Oracle clients to pay only for the functionality they need. Unfortunately, these options and packs can form a dangerous licensing trap.

Unlike most software vendors that distribute add-on components via separate install media or require explicit license codes to activate add-on features, Oracle offers its options to customers a little differently. For an Oracle database, nearly all of the available separately licensable options and packs are installed and enabled by default. In addition, most of these features are difficult, if not impossible to disable. Oracle has made these extra features easy to use unintentionally. Oracle software does nothing to prevent users from accidentally invoking unlicensed add-on functionality. But it zealously records all uses of these features. Couple this with the likelihood of an Oracle audit and many organizations discover that they have unknowingly accrued additional Oracle licensing liability. It is easy to more than double the licensing cost of an Oracle database with add-on options and packs. Extra options and packs can represent a significant source of extra cost if not carefully controlled.

It is also possible to invoke most of these options with an Oracle Standard Edition database even though Oracle's own licensing rules prohibit the use of the add-on options and packs with Standard Edition. Thus, an organization that inadvertently uses an option or pack on Oracle Standard Edition may encounter an awkward situation during an Oracle compliance audit. House of Brick has seen Oracle demand organizations buy Enterprise Edition licenses in addition to the specific options and packs to resolve this kind of licensing deficit during audit.

Accidentally stumbling into Oracle database feature usage from the add-on options and packs present a significant risk to an organization. While this risk can be partially mitigated by thoroughly training database support staff, House of Brick has seen plenty of incidents where well-trained staff inadvertently create a licensing liability by innocently invoking a feature from an add-on option or pack. House of Brick offers automated monitoring software and a license managed service to help our clients avoid these risks.

Updated Contract Terms

Staying on top of Oracle licensing can be tricky, requiring careful study of contractual documents to separate Oracle's marketing claims from contractual truth. It gets much harder when organizations realize that Oracle has the habit of updating contract terms over time. So not all their subsequent license purchases are on the same terms as their original licenses. As organizations experience organic growth in the scope and scale of their IT organizations, it is natural to incrementally acquire additional Oracle licenses. A call to an Oracle sales representative can typically get an order moving quickly.

What many organizations do not notice, however, is that the ordering document and invoice that Oracle sends often contain amended contractual terms. Acceptance of the order and submission of payment to Oracle constitutes agreement with the amended terms for the new licenses. Oracle will often offer a consolidation of all existing licenses onto a single order in conjunction with an incremental purchase, which they purport is for simplifying and synchronizing support renewals. While it does accomplish this, it also means that if the ordering document contains updated terms and conditions, then all Oracle licenses

owned by an organization are now under these newer terms and conditions. As a slight variation on that tactic Oracle sales representatives may offer to cancel and replace existing licenses onto a single ordering document for zero additional cost in order to consolidate licensing and ensure all support renewals are on a single invoice. Just as in the aforementioned consolidation exercise around incremental purchase, these cancel and replace exercises usually come with new contract terms in the ordering document.

Oracle has used these consolidation tactics over time to bring clients with older master agreements into accord with newer and more restrictive policies, counting on the fact that organizations are less likely to scrutinize incremental orders or consolidations that have little or no incremental cost. As Oracle's contract terms have universally become more restrictive and less customer friendly over time it is important to stay on top of any updates or amendments offered in order to maintain compliance and avoid trading away favorable terms and conditions for less favorable ones.

Oracle Traps and Risks: Organizational Innovation and Agility

The Oracle Change Fear Factor

In 2018 House of Brick was approached by a client, a major personal care product and dietary supplements company, to help move them to the cloud. They had been building their core mission-critical applications on Oracle for over a decade and discovered, as so many organizations do, that their Oracle database layer had grown so fragile and high maintenance that they were almost afraid to touch it at all. Years of deferred upgrades coupled with a loss of tribal knowledge from staff turnover resulted in a situation where the risks and costs of any potential changes were so high that it was alarming to contemplate any project to modernize.

This scenario represents an excellent example of an Oracle-related trap that snares many organizations. Complex Oracle solutions can easily become the tail that wags the dog if an organization does not take a disciplined approach to integrating them properly and strategically ensuring that their architecture, interdependencies, and operation are accounted for in overall IT strategy.

Oracle databases are notoriously fragile, high maintenance, and high touch when compared to other pieces of an application stack. What is worse, the database layer is typically the foundational layer and single point of failure for one or more applications. So, maintaining an Oracle database couples high complexity tasks with a high risk for bad outcomes that can lead to downtime in multiple applications.

For mission-critical databases expected to have 24/7/365 operation and extremely high uptime, the risk problem is far greater. The combination of high risk and high complexity requires specialized personnel while other teams across an IT organization must accommodate these Oracle risks into their planning and schedules to avoid impacts to critical infrastructure.

This leads to an effect that House of Brick calls the Oracle Change Fear Factor, where organizations move at a glacial pace for any initiative, enhancement, or maintenance that might involve an Oracle database because of the perception of extreme risk. This glacial pace eventually infects nearly all projects because databases are so central to solution stacks that it is nearly impossible to avoid them. As velocity slows throughout an IT organization, productivity is reduced, costs rise, and innovation is compromised.

Oracle Database Operational Cost Trap

Oracle releases quarterly patches for their database software mostly for security update purposes. While frequent security patching is widely seen in the IT industry as a best practice, IT organizations typically view

Oracle database security patches with more than a little trepidation. This is mostly due to past incidents of Oracle introducing behavior changes via security patches that impact application performance or functionality. For a mission-critical database, it is dangerous to deploy untested patches that might introduce behavioral changes into a production environment.

The proven means to combat these risks is testing. Building a rigorous testing plan around every patch will significantly reduce the risk of unforeseen outcomes; but it comes at a steep price. Such testing strategies require a test environment identical to the production environment to be provisioned and maintained. It also requires staff to create a dedicated process to test all patches for a database platform prior to production deployment. The costs in environments, licensing, and personnel effort can stack up quickly and make Oracle database estates even more costly to maintain.

Even after an organization comes to grips with a good Oracle patching strategy, other critical maintenance tasks will require constant attention. Backup and recovery strategies to protect key data, high availability and disaster recovery solutions, database storage and I/O growth, SQL statement tuning, and SQL statistics must all be monitored constantly. A failure in any of those areas could lead to a loss of business-critical data. Solutions must be designed, implemented, monitored, tested, and adapted over time by specialized staff. In almost all cases, Oracle databases require database administrators to use Oracle-specific software to handle these functions as opposed to tying into larger organization-wide general-purpose solutions already implemented for other portions of the solution stack.

All of this adds up to a considerable operational expense and one of the larger hidden costs to operating an Oracle database solution. Understanding the nature of these costs and the challenges that lead to them is vital for devising a cloud strategy.

Organizational Silo Trap

One of the more subtle challenges organizations face with any technology as complex as Oracle's relational database is an organizational silo. As noted earlier, a complex piece of technology like an Oracle database requires specialized database team to deploy, operate, and maintain databases. After establishing such a database team, usually completely distinct from traditional development or IT operations teams, there is a very real danger of forming an IT silo.

A silo, in the traditional organizational meaning of the word, is a team that does not interoperate with other teams effectively due to differing priorities, tools, and processes. IT silos create a drag on efficiency and speed for any IT organizations. IT professionals might encounter a database silo from any organization that might say: "We can't release on time, the Oracle DB team is busy with a Priority 1 issue" or "The Oracle DB team said they would want two months to regression test if we refactored the stored procedures, so we are dropping this feature as it can't make the project deadline." Silos are inefficient at best. They can lead to turf wars and blame game exercises that leave employees more concerned about competing with internal teams than in achieving larger organizational goals.

Database teams are especially prone to many of the negative outcomes of silos. The database team is typically a small team of specialists who must interoperate with every other IT function or team. However, they also typically use separate tools and workflow processes that are unique to their specialized domain. They maintain complex technologies that form a core central component of one or more application stacks, usually business critical ones, and tend towards extreme aversion to change or risk. This is the classic recipe for the "No" group that tries to resist any IT change or improvement. The kind of group that other teams work around, even engaging in shadow-IT practices, because it is easier than working with them.

Forming a database specialist silo in an IT organization can drag down velocity and efficiency of the IT organization, sometimes quite dramatically. In 2019 House of Brick helped a major west-coast municipal

government craft a cloud strategy for their large estate of Oracle databases. The client reported that the Oracle database estate was a major expense that was getting impossible to maintain. Working with the client, House of Brick identified the technical hurdles that made the Oracle databases such a burden. To everyone's surprise, the major findings of the investigation were as much organizational as they were technical.

Over the years this organization had leveraged outsourced database specialists to maintain their databases. The outsourcing firm was doing a good job of maintaining the systems and keeping them operational and performant but viewed any request for a change or upgrade in the database platforms as an out-of-contract project that required separate business and financial arrangements.

As a result, other parts of the IT organizations viewed any database change as too painful to contemplate. It was the worst database silo House of Brick had ever seen with some databases running versions of Oracle that had been out of support for almost a decade. Many databases were up and running long after the associated applications were retired, just to preserve reporting access to legacy data. This mess developed because maintaining the costly and inefficient status quo was easier than trying to negotiate a final archival project with the outsourced database support organization

House of Brick helped this client, and many others, consolidate their databases into a far more efficient footprint. It is unfortunately common for specialist Oracle database teams, whether outsourced or in house, to become an impediment to organizational velocity and efficiency.

Organizational Overhead Trap

One of the hidden costs of Oracle licensing is the overhead of managing the contracts and licenses. While this function, usually shared by IT and procurement or finance departments, is true of any software vendor, the complexities and high ongoing costs of Oracle software licenses coupled with Oracle's reputation for aggressive audit practices demand extra attention. This creates an ongoing drag on an organization as they must either devote time and attention to maintaining compliance with complex Oracle licensing rules or face significant costs if caught unprepared in an audit.

Staying on top of all these aspects of dealing with Oracle requires vigilance, knowledgeable staff, and an investment in time and effort that could detract from more strategic pursuits.

AVOIDING THE ORACLE TRAPS AND RISKS

With the cost traps and risks fully detailed, the mind naturally turns to the next question: How can an organization avoid these traps during an AWS migration? House of Brick has a wealth of optimization strategies specific to the AWS environment that can help organizations plan a cost-effective migration that avoids them all.

AWS Migration Targets for Oracle Databases

When migrating an Oracle database to AWS the first key decision organizations face is whether to keep the database portion of the application stack on an Oracle relational database or look to another database engine to manage their data. For application stacks such as Commercial off-the-shelf (COTS) software that cannot be easily ported to other database engines, the second key decision very quickly becomes whether to deploy Oracle on the EC2 compute service or take advantage of the Amazon RDS for Oracle, a managed database service. Many organizations agonize over these choices. So, before going into the details of each choice and how they assist in avoiding Oracle costs, it is important to have at least a high-level understanding of the pros and cons of the three options.

Option 1 – Oracle on EC2

The first option, a migration to customer-managed Oracle on EC2, is the most conceptually familiar. EC2 is the popular cloud virtual compute service offered by AWS. An EC2 instance can run any common operating system, has a great deal of flexibility in sizing and configuration, and offers the greatest operational flexibility for managing the system. Think of an EC2 instance as functionally identical to an on-premises server or virtual machine. This means all aspects of operating an Oracle database on an EC2 instance use the same familiar practices as an on-premises environment. The provisioning, maintenance, and monitoring requirements are all the same. EC2 offers all the flexibility as an on-premises Oracle installation and can support virtually any kind of customizations or configurations that an organization might be using.

The only drawback to using EC2 as a target for Oracle database migration is that all the traditional activities and risks of a customer-managed Oracle stack still exist. Provisioning, configuration, patching, upgrades, and backups are all still tasks that skilled staff must devote time and attention to performing. Of all the AWS migration targets for Oracle, the EC2 customer-managed target will most likely cause an organization to suffer to some degree from the Organization Silo Trap, the Operational Cost Trap, and the Oracle Change Fear Factor.

Option 2 – Amazon RDS for Oracle

The second option is a move to a managed Oracle service, which in AWS means using the Amazon Relational Database Service (RDS) for Oracle. Amazon RDS for Oracle is quite different from EC2 in that it is a managed database offering that provides a method to support Oracle databases while avoiding all the overhead of provisioning and operating an Oracle database. This managed service approach does require operating within the range of AWS-offered Oracle versions and configurations but that is a concern only for a very small minority of organizations that utilize specialized or exotic Oracle configurations. The upsides of using a managed database service are quite significant and are given a full treatment in the next section of this paper.

Option 3 – Open-Source or Cloud Native Engine Migration

The final option to consider is migrating away from Oracle entirely. The most popular alternative, due to having the closest feature parity with Oracle database, is Amazon Aurora with PostgreSQL compatibility. House of Brick has also seen clients migrate Oracle databases to high-performance non-relational database such as DynamoDB or Neptune. Migrating away from Oracle to an open-source or cloud native

engine offers significant cost savings and potential performance gains, as well as all the already mentioned advantages of a managed service.

The only drawback that prevents House of Brick from recommending this approach for all Oracle databases being migrated to AWS is the necessity of ensuring application compatibility. For COTS applications, support for different database engines may not be available from the application vendor. When considering legacy application stacks developed in-house that are not intended to remain supported beyond the short term, the up-front cost and effort to refactor them to support a different database engine might not be justified by the potential savings. For all other Oracle databases, House of Brick suggests moving to an open-source engine such as Amazon Aurora with PostgreSQL compatibility. The advantages of this migration target are explored in more detail later in the paper.

Making the right decision on migration targets is a major factor driving the possible cost savings, but it is not the only opportunity. House of Brick has helped many clients that reconsidered their initial migration choices. It is important to understand that the flexibility of the AWS environment makes it possible to change course after a less than ideal initial migration strategy and still realize significant cost savings.

Amazon RDS Advantages for Oracle

Using Amazon RDS to Avoid Oracle Cost Traps

According to IDC , organizations can reduce costs by around 200% when using Amazon RDS for Oracle as their Oracle database environments[3]. These impressive savings come from the many Oracle cost traps that Amazon RDS can help an organization avoid.

A big advantage of turning over management of a database to a managed service provider is the ability to avoid the headaches of day-to-day relational database management. This directly addresses the Oracle Change Fear Factor and the Oracle Database Operational Cost Trap. When using Amazon RDS for Oracle, AWS performs the tedious configuration and operational management for key activities like patching, backup, and high availability. It also performs them to a higher standard than most organizations can do on their own. Because AWS operates RDS at a vast scale, they can invest more time and engineering effort in these activities and still provide their results to end users at attractive costs.

With Amazon RDS, database provisioning and patching are simple and streamlined tasks that do not require specialized personnel. This does not eliminate the need for Oracle specialized personnel to perform tasks like schema design and query tuning. But it does free them from routine database operation tasks. This allows many DBA departments to diversify their strategic focus and reduce the "no" factor that Oracle database teams are notorious for. Just freeing up the specialized personnel from the day-to-day grind helps avoid the Organization Silo Trap.

As discussed, a rigorous testing strategy does not guarantee every negative behavior variance introduced by upgrades or configuration changes will be caught in time. The only guarantee of a rigorous testing strategy is that it will incur cost for compute, licenses, and personnel time. This makes Amazon RDS a fantastic option for avoiding both the Oracle Change Fear Factor and Oracle Organizational Overhead Trap. Relying on Amazon RDS for Oracle to test patches and configurations relieves organizations of the need to do it on their own. There is still a need for full testing on major Oracle upgrades of course, to satisfy application-level compatibility concerns. Amazon RDS for Oracle still removes most of the effort and concern around such testing. With Amazon RDS for Oracle, a new upgraded environment for testing can be provisioned in minutes. At the end of the day, organizations that turn over patching and upgrades of their

[3] https://pages.awscloud.com/GC-600-business-value-rds-ebook-confirmation

Oracle database solutions to Amazon RDS get a polished and stable configuration with excellent performance.

Taking Advantage of Amazon RDS for Oracle Standard Edition 2 and License Included

Oracle Database Standard Edition 2 is one of the most well-kept secrets of the Oracle product portfolio. It is nearly 66% cheaper than the Enterprise Edition, and Oracle has traditionally positioned it as an entry-level product while counting on enterprise customers upgrading to the Oracle Database Enterprise Edition to get the kind enterprise features required for business-critical Oracle workloads. However, the Amazon RDS managed service provides a wealth of enterprise features that fill the gaps Oracle deliberately left in the Oracle Standard Edition 2 product. This raises the intriguing possibility that Oracle Database Standard Edition 2 plus Amazon RDS for Oracle together can create a combination of technologies to run mission-critical Oracle workloads in AWS and avoid the costs of Oracle Database Enterprise Edition.

The Oracle Database Enterprise Edition and the Oracle Database Standard Edition 2 offer identical features in most areas, but the Enterprise Edition has a few key exclusive features in the areas of performance, security, and high availability. The omission of those features from the Standard Edition 2 means that traditionally many enterprise applications had no choice but to pay the higher costs of Oracle Database Enterprise Edition to get features required for mission-critical workloads. When moving to AWS, however, an intriguing alternative exists. The Amazon RDS for Oracle service features in those areas can supplement the gaps in Oracle Standard Edition 2's feature set. The combination of the Amazon RDS service and Oracle Standard Edition 2 creates an enterprise-class database solution for a fraction of the price of the Oracle Database Enterprise Edition.

Beyond the technical advantages of the Amazon RDS For Oracle, the Standard Edition 2 offering becomes even more compelling when considering that not only the compute and storage are available from AWS on a pay-as-you-go basis but also the temporary rights to use the Oracle license. With Amazon RDS for Oracle there is a License Included offering for Oracle Standard Edition 2 that allows an organization to deploy and operate Oracle Standard Edition 2 without purchasing Oracle licenses. Not having to track Oracle licenses, maintain rigorous compliance standards, or worry about a compliance audit from Oracle is a fantastic advantage and generates significant operational savings by eliminating the overhead around all those activities.

With all this promise in mind, a fuller exploration of each advantage is warranted.

Performance

One of the main limitations of the Standard Edition 2 product is that the database is technically limited by Oracle to 16 simultaneous threads of execution with no parallelism. This limitation is far less severe than most database administrators initially assume, as modern high-performance CPUs can do quite a bit with 16 simultaneous threads of execution. House of Brick has seen Oracle Database Standard Edition 2 systems perform quite well in online transaction processing workloads, easily handling over a hundred queries per seconds. And there are public TPC-C benchmark results demonstrating Oracle Standard Edition execution over 1,000,000 transactions per minute when run on powerful hardware[4]. As a general rule, database performance is limited by the speed of disk storage more than the availability of CPU cycles. When coupled with fast flash storage even Standard Edition 2 Oracle databases can deliver impressive performance.

[4]https://www.tpc.org/tpcc/results/tpcc_result_detail5.asp?id=111120802

Security

Oracle typically reserves encryption features such as Transparent Data Encryption (TDE) for only the Enterprise Edition versions of their product. TDE is not included in the Oracle Database Enterprise Edition offering directly, but it is available as an additional cost option for Enterprise Edition called the Advanced Security Option (ASO). Organizations transitioning to the Amazon RDS for Oracle service are usually delighted to find that the standard Amazon RDS feature set offers full encryption at rest for no additional charge. For Amazon RDS for Oracle SE2 instances full database encryption, either with AWS managed keys or customer managed keys, is now a simple checkbox either at creation time or as a modifiable option that can be implemented later. Additionally, key management is streamlined through the AWS Key Management Service (KMS) so rotating and managing encryption keys on Amazon RDS is a much simpler exercise than using the native Oracle tooling for ASO.

High Availability

With on-premises deployments, the Oracle Database Enterprise Edition option Real Application Clusters (RAC) is the most used solution to provide high availability to a mission-critical Oracle database. Amazon RDS for Oracle has no offering that includes RAC. But the Amazon RDS for Oracle service provides HA features for critical databases that require them. In Amazon RDS for Oracle the Multi-AZ deployment feature exists to provide high availability to databases. This feature creates a copy of the Oracle database in another availability zone of the same region with duplicated storage and compute resources. In addition, this feature provides smooth automated failover to the duplicate so in the unlikely event of a system-down event for the primary database, the duplicate can take over all processing responsibility in minutes. It is worth noting that the duplicated compute and storage resources means that the compute, storage, and licensing costs for such a solution are doubled. However, even doubled, an Amazon RDS for Oracle Standard Edition 2 deployment is still less costly in most cases than a non-highly available Oracle Database Enterprise Edition deployment.

Disaster Recovery

Business-critical workloads require disaster recovery (DR) features to prevent large-scale events that bring down entire datacenters from stopping an organization from functioning. In on-premises environments, Oracle platforms typically achieved this with a combination of off-site backup storage for Standard Edition databases and the Data Guard feature of Oracle Database Enterprise Edition to maintain a synchronized copy of the data in another datacenter. These types of DR tolerant architectures typically took careful setup and monitoring to ensure proper functioning.

After moving to AWS both kinds of functionality are available to Enterprise Edition and Standard Edition databases utilizing Amazon RDS for Oracle tools. Doing these same things in Amazon RDS is even easier and less costly than in on-premises environments. Off-site backups are easy to accomplish by storing Amazon RDS for Oracle database backups in S3 storage in another account to ensure that no disaster can wipe out key data backups. Amazon RDS for Oracle can even automate this with the Amazon RDS Cross-Region Automated Backups feature.

The ability to maintain a near-real-time copy of an Oracle database in another region has long been reserved for the Oracle Database Enterprise Edition, either via Oracle DataGuard or Oracle GoldenGate replication technologies. These technologies were useful in situations where using backup based DR strategies did not support the desired recovery point objective or recovery time objective. In the AWS environment, when using Amazon RDS for Oracle with Standard Edition 2 the AWS Database Migration

Service (DMS) can be used to replicate data from an Oracle database into another similar database in another region to achieve excellent data replication and DR capabilities.

Licensing Advantages

One of the most compelling points about the Amazon RDS for Oracle Standard Edition 2 License Included option is the license. Being able to acquire rights to use Oracle database software on a pay-as-you-go basis directly from AWS removes all the headaches of Oracle software licensing. This avoids the need to negotiate prices with Oracle, budget in years of costly maintenance payments, or plan for the possibility of Oracle license audits. No longer having to deal with Oracle licensing costs and Oracle support renewals are the primary benefits for using license included Amazon RDS for Oracle Standard Edition 2 License Included as this completely evades the Oracle Licensing Cost Trap.

Being able to use an Oracle Standard Edition 2 license on a pay-as-you-go model allows operational flexibility to pay for the license only when needed. Oracle licensing contracts typically require long-term up-front commitments from organizations including an upfront one-year prepayment. The ability to grow or shrink both database footprint and licensing footprint as needed is advantageous for organizations using Amazon RDS for Oracle License included and boosts not only cost savings but organizational agility as well.

Migrating Oracle to Amazon Aurora

Samsung recently reported a 44% reduction in monthly costs to operate their database estate[5]. Their technique for accomplishing this impressive reduction was hardly revolutionary. They simply stopped using Oracle Database.

One of the most straightforward solutions to avoid the costs of using an Oracle database is to no longer use Oracle's database products and instead migrate to an open-source database engine. While conceptually the obvious way to avoid Oracle costs, most organizations shy away from refactoring critical applications as they anticipate a great deal of effort and technical risk to be associated with such an initiative. Refactoring requires technical work as connected applications require modifications to use a different database engine and may not be possible with COTS applications that only support Oracle database as a back-end datastore. In scenarios where it is possible, the potential cost savings can be extreme.

House of Brick has helped multiple clients migrate their Oracle databases to Amazon Aurora as part of a cloud migration.

The most popular target is Amazon Aurora with PostgreSQL compatibility as it offers many of the same general capabilities as the Oracle database engine, but other targets may make sense as well depending on the nature of the application stack. There is some work involved to refactor a solution stack to support a different database engine, though it is generally a lot less than most organizations fear. Application SQL, application connectivity, schemas, and stored procedures may all require refactoring.

While this prospect can seem daunting and high-risk, House of Brick has observed that most clients vastly overestimate the effort and testing required to migrate a solution stack as part of a move to the cloud. AWS has purpose-built tools, such as the AWS Schema Conversion Tool (SCT) and AWS Database Migration Service (DMS), that automate away most of the drudge work from such a migration making it feasible and

[5]https://aws.amazon.com/solutions/case-studies/samsung-migrates-off-oracle-to-amazon-aurora/

reducing risk. The initial effort to migrate is usually easily justified as the savings from releasing Oracle licenses and terminating support payments that quickly lead to a positive return on investment.

Once migrated to an open-source database engine such as Amazon Aurora PostgreSQL, Oracle licensing costs can be reduced and offer greater operational and cost flexibility. With no need to worry about licensing compliance, teams are free to embrace more efficient workflows and gain all the benefits of DevOps practices. With the flexibility of pay-as-you-go expense models in AWS, it is easy to achieve license savings and operational savings with Aurora, as scale-up and scale-down in response to demand are quick and easy.

Migrating an Oracle database to an open-source engine on a managed service avoids every one of the cost traps that this document focuses on. It is a powerful tactic for drastically reducing licensing costs. And if done correctly, it can achieve those savings without sacrificing features or performance.

Rightsizing an Oracle Environment in AWS

Regardless of which migration target an organization chooses for their Oracle databases there are significant advantages to rightsizing an environment. Under provisioning traditionally leads to scaling pain and costly business impacts as mission-critical system struggle to cope. Over provisioning incurs extra costs for no marginal benefits as idle capacity goes unused. When right sizing a database in a pay-as-you-go environment like AWS, all the traditional sizing methodologies apply for picking the right compute and storage dimensions for a platform. But there is also the additional wrinkle of rightsizing the operational windows. Getting both of those aspects right helps avoid the Oracle Licensing Cost Trap.

The Case for Rightsizing

An important factor for constraining Oracle costs in the cloud is avoiding over provisioning. Oracle database administrators have long indoctrinated habits of over-provisioning resources that prove difficult to break when moving to AWS. In an on-premises environment, over-provisioning has traditionally been an understandable defensive strategy to guard against sudden growth in transaction traffic to databases. In an on-premises environment, where timeframes to acquire new server capacity can be measured in months, most Oracle database teams learned the hard lesson to overprovision whenever they had an opportunity. Having a key database fail because business (and thus database traffic) grew by 25% in a stellar quarter would reflect badly on teams tasked with maintaining the integrity of business-critical systems.

In AWS, additional storage or compute capacity is only a few clicks away. Overprovisioning and committing to expenditures now to potentially ward off growth problems later makes no sense. Yet the habit proves difficult to break for DBAs. House of Brick has worked with many clients to aggressively pursue every opportunity to reduce their storage and compute footprints during migration instead of migrating databases on a like-for-like basis.

Under-utilization of servers in a data center that has been understood and written about for years. A quick look at the industry average resource consumption rates will show that the average on-premises database is vastly underutilized on most resources, even having cushion to spare during peak periods. Wired magazine was calling attention to this phenomenon as far back as 2012 (https://www.wired.com/2012/10/data-center-servers/) . While exact numbers are difficult to come by, most experts conclude that the average data center server utilization is only in the 6%-12% range, With only 6%-12% usage the implication is that servers are overprovisioned by approximately 800% as a matter of

course, which serves as a stark demonstration for how much overprovisioning is an ingrained behavior in IT organizations.

Taking that same overprovisioning mindset to the cloud results in extensive extra costs for no benefit. Not just expenditure on compute and storage capacity, but database licensing costs can also grow out of control as overprovisioned database systems incur extra licensing burdens. Recall that the number of vCPUs determines how many Oracle processor licenses are required for a RDS or EC2 instance in AWS. This means keeping an Oracle platform in AWS sized to the minimum required vCPU count is a practical tactic to minimize the use of costly Oracle software licenses.

Imagine a scenario with an Oracle database on EC2 that could perform acceptably at current traffic levels with a 4 vCPU r5.xlarge instance. Now imagine that same system was originally overprovisioned per the aforementioned on-premises averages. , and thus was 800% over-provisioned. Such a system would end up as an as a 32 vCPU r5.8xlarge by database administrator staff that is accustomed to overprovisioning as a matter of course.

This increases the compute costs over a year from $2,207 to $17,660, using published on-demand pricing for the US-EAST-1 region. A premium of a bit more than $15,000 annually seems potentially reasonable for having a buffer of extra capacity on a critical system. Many organizations might not consider that excessively wasteful but looking just at the compute costs misses a big contributor to total costs. Oracle licensing costs easily eclipse compute costs. Looking only at the compute cost differences is missing the biggest source of cost.

When viewed from the licensing perspective, the r5.8xlarge in our example requires 16 Oracle processor licenses whereas the r5.xlarge requires only two. At list price, a single Oracle Database Enterprise Edition processor license costs $47,500 plus $10,450 annually in SULS. But that is a bit misleading as it is rare for organizations to pay full list price. To give a more realistic way of computing the costs of Oracle licensing, House of Brick prefers to apply a reasonable discount from list price and then calculate a total Oracle cost by summing the license purchase price and several years, typically three or five, of support costs. Once that sum is divided by the number of years an annualized price can be derived. Such annualized prices are a good metric to use when determining how much an individual system contributes to the overall Oracle Database licensing costs for an organization.

Assuming a scenario where an organization negotiated a 25% discount from Oracle list prices, the three-year annualized price of licensing and support for a single Oracle Database Enterprise license is $19,910.18. When factoring in these licensing costs, it is easy to compute that the licensing cost difference between the r5.8xlarge and r5.xlarge is $318,562.88 vs $39,820.36 annually. Overprovisioning that leads to around $15,000 in extra compute costs may seem reasonable, but overprovisioning that leads to more than $250,000 of extra licensing cost annually is completely indefensible. Even if provisioned with existing licenses, an r5.8xlarge could easily cost many tens of thousands dollars more per year more to operate than a r5.xlarge just due to the Oracle SULS payments.

While this was only a hypothetical example and may seem extreme, House of Brick has seen real-world versions of this same scenario play out at many clients. Rightsizing an Oracle platform in AWS can generate significant cost savings. Failure to size appropriately can lead to significant unnecessary licensing expenditures.

Rightsizing Techniques for Oracle Databases in AWS

To right size an Oracle system in either EC2 or RDS, and achieve significant cost efficiency, it is important to have database specialized staff evaluate key metrics for vCPU requirements, memory requirements, and storage requirements to arrive at the optimum configuration. The House of Brick best practice for this exercise is to size systems to the traffic they serve in the immediate term, not the longer term. Long-term growth is easy to accommodate by utilizing the elasticity of the AWS environment to scale systems on-demand when they need to grow. Rather than trying to precisely forecast system growth over time, in AWS it is far better to simply monitor systems and provision extra capacity shortly before it is needed.

Oracle databases include two potential rich sources of metrics data for sizing. As long as the on-premises databases are instrumented with either Oracle Statspack or Oracle AWR, the right metrics should be available. It is important to note that the Oracle AWR feature offers richer metrics but is part of the separately licensed Oracle Diagnostics Pack. This Diagnostics Pack is available only as an add-on for Oracle Database Enterprise Edition, so organizations should utilize the Statspack metrics for sizing unless working on an Enterprise Edition system specifically licensed with Oracle Diagnostics Pack.

While AWR and Statspack data is rich with complex statistics, the key metrics for a sizing exercise are easy to extract. House of Brick has a proven approach to sizing databases in the cloud, which consists of the following steps:

- Look at database CPU utilization and I/O utilization over time and calculate a 95th percentile
- If migrating a database to the cloud from an older on-premises server where the CPU architecture is dissimilar to the target, use published SPECInt 2006 or SPECInt 2017 benchmarks, particularly the Integer Rate Results metrics, to calculate a conversion factor
- Multiply the conversion factor, if one was required, by the 95% percentile observed CPU usage to calculate the final CPU requirements
- Once the appropriate CPU targets are established, determine the needed IOPS and I/O throughput to meet the 95th percentile target
- Calculate memory targets by using the V$SGA_TARGET_ADVICE view.

The combination of the calculated CPU target and the conversion factor should gauge the CPU usage requirements accurately, which is the most important architectural factor to size correctly. Using these factors should let an organization quickly identify the minimum instance sizes and families that may serve the workload. However, it is often the case that CPU requirements will suggest a smaller instance size whereas RAM or I/O requirements might suggest a larger instance size. When faced with this dilemma many organizations that House of Brick has worked with experience a temptation to fall back into overprovisioning. Happily, there is another way forward using the Optimize CPUs feature.

Optimizing Oracle Licensing with Optimize CPUs

Oracle has special licensing rules for the cloud, as mentioned previously, and they allow an organization to license Oracle software products on a vCPU basis. In an on-premises environment licensing is calculated on the number of physical processor cores in use, or on a combination of named users and physical processors. In AWS and other clouds, the required Oracle licenses costs are based on virtual CPUs.

Translating between the two can sometimes seem like the cloud database solutions will require double the licensing, as Oracle is fond of pointing out. But that is rarely the case in House of Brick's experience. Recall that overprovisioning is a frequent source of waste and most real-world databases are limited by available disk I/O bandwidth as opposed to available CPU cycles. It is quite likely that databases can be migrated to AWS with careful rightsizing and achieve no increases in licensing while avoiding any adverse effects on overall performance.

One of the most powerful tools available in AWS for rightsizing an Oracle database is the AWS Optimize CPU feature. This feature allows provisioning larger EC2 or RDS instance shapes with reduced virtual CPU counts. The reduction in CPU capacity does not lead to a direct reduction in compute costs, which initially makes it sound like an odd feature to consider when optimizing costs.

Recall in the previous section on overprovisioning the hypothetical example listing the compute and licensing cost differences between a r5.xlarge instance and r5.8xlarge instance. The difference in compute costs was a few thousand dollars but the difference in licensing costs was a few hundred thousand dollars.

When deploying in AWS Oracle Database licensing costs are typically an order of magnitude higher than compute costs.

Keeping a focus on minimizing licensing costs with Oracle requires maintaining a laser focus on minimizing allocated vCPUs. The Optimize CPUs feature for RDS or EC2 can be an effective tool for reducing the licensing requirements of a cloud database while still taking advantage of the memory and I/O advantages of larger instance shapes.

House of Brick has heard from many clients that Oracle sales representatives or auditors have tried cast doubt on the Optimize CPUs feature of RDS and EC2, claiming that regardless of the provisioned vCPU counts an instance must still be licensed as if all the vCPUs were enabled. There are not, to the best of House of Brick's knowledge, any terms in the Oracle software licensing contracts that support this claim. Indeed, House of Brick has rich experience in assisting clients with Oracle compliance audits and have seen multiple clients successfully defend the idea of disabling physical CPUs, processor cores, or virtual CPUs to limit licensing liability.

Optimizing Operational Windows

With AWS, the unique nature of pay-as-you-go Amazon RDS environments makes rightsizing techniques available that have never traditionally been available in an on-premises environment. Database environments for applications that are not 24/7 in nature can have their runtime windows rightsized to avoid the costs of operating them outside of their needed operational time. This can be a surprisingly powerful technique for reducing Oracle database compute costs on RDS or EC2.

In AWS, an Amazon RDS or Amazon EC2 instance incurs no cost beyond storage when it is offline. Compute costs are zero. For an organization like a municipal government, with defined hours of operation, that means that most databases were completely unused on evenings and weekends. Utilizing AWS Lambda it is quite simple to craft some scheduled tasks to shut down databases when not needed and then start them up again when they are. Amazon even provides a solution, AWS Instance Scheduler, which provides a template for doing just that. For a database that is only needed for nine hours a day instead of 24 hours a day, leaving it offline when not needed can lead to considerable savings. Typically, this is in the range of 50% to 60% as most databases do need to be online outside of core business hours on some occasions for reporting jobs or maintenance jobs or the like.

Recall the municipal government that House of Brick was assisting in sorting out their organizational challenges that led to a great many legacy databases being left online, without upgrades or patches, long after the related applications has been retired? House of Brick was able to help them reduce their Oracle costs in the cloud with a variety of strategies and one of the most impactful was rightsizing the databases for their operational window. Most of their databases support departmental-level applications that were utilized only during regular local business hours on weekdays. By leveraging Amazon RDS for Oracle

License Included it was possible to zero both compute and licensing costs when the systems were not in use. This, along with consolidation and effective archival of legacy data, allowed House of Brick to give the client a path to reduce cloud spend on Oracle databases by over 40%.

Consolidate and Archive Efficiently

Consolidation as a tactic to reduce Oracle database usage is surprisingly effective in a variety of scenarios. Sometimes the portfolio of Oracle databases operated by an organization will contain complementary databases that share similar load profiles or back the same application. In most cases they were created as separate databases initially for sound operational reasons, perhaps because the servers or storage available at the time could not handle the load of both databases on one server. Such decisions, when revisited years later, often do not hold up when considering the high-performance compute and storage capacity available in AWS. If, upon review, such datasets could easily co-exist in the same database, then combining them is quite easy using Oracle tools. Performing such combinations allows an organization to take fuller advantage of costly Oracle licenses.

It is not uncommon for databases to become obsolete, as new applications are developed to replace older ones. However, there is always an aversion to deleting old data. Amazon RDS is an ideal use-case for these older databases that need data preservation but are not actively used by current applications. Migrating an archival database to Amazon RDS, particularly to an open-source engine, and then only bringing it online when needed leads to significant cost savings while still leaving key data fully available.

To continue the story of House of Brick's municipal client with a great many legacy databases, one strategy that helped to rein in their Oracle database costs was finding a good final archival strategy for their legacy databases. They had a variety of legacy Oracle databases that had been kept online despite the associated applications being retired as there was a desire to be able to perform ad-hoc queries on legacy data. Operating Oracle databases on a 24/7 basis to serve only a few ad-hoc queries a week was quite wasteful considering the high licensing costs of those databases. House of Brick was able to help reduce the costs for legacy databases by more than 80% by creating an Aurora PostgreSQL cluster to serve as a centralized archival database. Using AWS tools such as Schema Conversion Tool (SCT) and Database Migration Service (DMS) the schema and data were copied from the original databases into the Aurora cluster and the original databases decommissioned. The central archival databases for these legacy data sets provided a cost-effective means to retire multiple Oracle databases while keeping all the important historical data available for ad-hoc queries.

Effective Strategies for Oracle License Wind Down

While this paper has made many references to ways to minimize the use of Oracle licenses to avoid the Oracle Licensing Cost Trap, translating reduced Oracle license usage into actual cost savings takes a bit more time and patience. Oracle software contracts seem complex, and the rules for terminating support on an existing license or terminating the license entirely contain a few traps for the unwary. Happily, with a little understanding of the Oracle licensing rules, the wind-down strategies are straightforward.

Overview of Wind Down Strategies

Oracle licensing wind down techniques can be grouped into two broad strategies. The first strategy, applicable to Oracle customers under an Unlimited License Amendment (ULA), is fairly simple due to the fact that every ULA House of Brick has ever seen contains a Total Support Stream clause. This clause, in straightforward terms, binds a customer to a perpetual minimum spend for annual SULS payments to Oracle. Thus the best strategies for Oracle customers with a ULA in place is to accelerate efforts to

migrate all Oracle databases to alternative database engines. Once that is completed, a complete cancellation of the product licenses listed in the ULA is required to achieve any savings.

The second strategy, available to Oracle customers who have acquired individual software licenses on a Processor or Named User Plus metric, is to reduce usage and then leverage that reduced license usage to reduce annual SULS payments to Oracle. The key factor to keep in mind for anyone trying to reduce Oracle annual payments after a reduction in Oracle software usage is the timing of annual support renewals. An organization may terminate support on Oracle licenses, or even terminate the licenses entirely, at any time with a simple notification to Oracle. Unfortunately, such early termination will result in no immediate cost reductions, as support payments are always made in advance and are non-refundable. Thus the challenge is to identify the proper steps that will allow an organization to turn reduced Oracle software usage into reduced annual support payments to Oracle.

Oracle SULS Reduction Strategies

The first opportunity for cost reduction comes at the next annual renewal invoice. Thus, any effort to reduce Oracle licensing costs must be carefully scoped over a multi-year timeframe to match reductions in usage to annual SULS renewals. A realistic goal for an Oracle license cost reduction project would be results in two to three years. Even the immediate and complete abandonment of Oracle software products, which is rarely feasible, cannot generate any licensing cost savings until the next support renewal date.

To plan for effectively reducing an Oracle licensing portfolio, it is important to keep in mind a few of the licensing rules for Oracle software. The key rule of import for reducing annual support costs is support repricing. As explained earlier, this rule states that if a customer terminates support or terminates licenses entirely from a particular order, Oracle may recalculate the support payment cost for the surviving items on that order to eliminate any originally negotiated discounts. Taking this repricing factor into account is critical to ensure that reductions in Oracle license usage turn into measurable cost reductions.

The combination of the License Sets rule and the support repricing rule makes it tricky, but not impossible, to terminate excess Oracle licenses and realize savings. The key to doing so is to understand not only the portfolio of Oracle licenses owned, but also to understand which licenses are contained in which orders. It is entirely possible for an organization to work that out on their own by scrutinizing the history of original ordering documents and invoices for purchases of Oracle licenses to see which licenses were originally purchased together on an initial order. It is also possible for an organization to contact their Oracle account rep and have that individual provide the information, though generally it requires asking several times. Oracle reps are sometimes reluctant to part with this information or misunderstand the request. Typically, a very direct request that very specifically asks for a listing of licenses that includes not just the original purchase price and renewal costs but also the “Order ID” field will prove effective.

Once the listing of orders is available, finding a cost reduction strategy is straightforward though often tedious. The list of excess licenses that will be freed by reductions in Oracle license usage should be compared to the list of orders to find, if possible, a single order or two that contain all those licenses. At that point, a bit of math with the Oracle global price list and the standard support rate (22% of purchase price) will indicate quite closely how the remaining items will be re-priced for support renewal purposes.

Note that the 22% annual SULS cost is itself subject to a form of compound interest, as Oracle reserves the right to increase the support payments from an order by up to 4% annually depending on the specific amount listed in an organization’s Oracle agreements. Generally, the older the order the greater the financial benefit from cancelling it. In an ideal situation all the licenses targeted for termination will be from the oldest order with the worst originally negotiated discount. Of course, this will need to be balanced with the

consideration that older licenses may have more favorable terms and conditions associated with them. More realistically, the list of orders will require scrutiny and comparison to the most recent SULS invoice to determine which orders can be advantageously cancelled.

As a final note, while it may sound counterintuitive it may sometimes be more cost effective to cancel an entire order of licenses, even if it contains a few products still needed. The needed products can then be re-purchased as new Oracle licenses. If done correctly, this strategy can result in reasonable savings, but it does take time to negotiate the potential re-purchase price with Oracle in order to arrive at a discount that will make the strategy successful.

To illustrate the interactions of all these factors, consider a situation wherein an organization is using Oracle Database Enterprise Edition licenses purchased 15 years previously at a 25% discount from full list price which was, in 2005, $40,000. The annual support payment for each of those $30,000 licenses, at the standard 22% of purchase price, would be $6,600 in the first year. However, with the compound nature of the annual Oracle uplifts the actual support costs for those licenses will have increased over time. After 15 years of a compounding 3% support cost increase, each of those licenses would cost $10,282.58 annually for SULS. At the end of that period terminating a license from that order would cause the support price of each surviving license to jump to $13,710.11 after Oracle performed the repricing discussed earlier. That is a steep price for annual support on a single license.

House of Brick often counsels clients paying high SULS rates for older licenses to inquire with Oracle about purchasing new ones and canceling the entire order of old licenses. Re-pricing is thus avoided, and the new licenses are not suffering from years of compounded SULS. To extend our example further, imagine that the organization retiring the 15-year-old licenses was able to negotiate the purchase of replacement Oracle Database Enterprise Edition licenses at a 40% discount. Each of these licenses would cost $28,500 and $6,270 for the first year of support. With a savings of over 50% annually per license on the SULS cost compared to the re-priced licenses, such a repurchase tactic can show a positive return on investment after only a few years.

Benefits of DevOps for Oracle Databases

Moving to the cloud offers excellent opportunities to integrate database specialists and technologies more tightly with the rest of the IT organization by leveraging the flexibility of AWS and DevOps practices. DevOps is a popular new paradigm in IT for taking maximum advantage of cloud environments. AWS defines DevOps thusly:

> *DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations to better serve their customers and compete more effectively in the market. (https://aws.amazon.com/DevOps/what-is-DevOps/)*

DevOps can offer organizations speed and agility by combining the development and operational teams, and break down the silos between different disciplines. This is a potent way of combating the Organization Silo Trap and the Oracle Database Operational Cost Trap by implementing more efficient practices.

DevOps Benefits

There are many benefits to adopting a DevOps model for the operation of Oracle databases. These include:

- Cost savings at both the compute resource and licensing levels.
    - Manually provisioned databases tend to stay around when not needed, as personnel wish to avoid repeating the provisioning and configuration effort should the DB be needed again. Having automatic provisioning tools encourages organizations to create databases on demand and be fearless in destroying them when no longer needed.
- Operational flexibility – new instances are quickly provisioned as needed, and easily removed when they are no longer needed.
- Operational consistency – anytime a process can be automated it guarantees that the resource provisioning will be identical every time.
- Increased security – all appropriate ACLs, security groups, users, and roles will be clearly defined, and the usage of automated pipelines and repositories guarantees that all of this is auditable over time.
- Reduced risk – in a DevOps model the use of automated pipelines and frequent, smaller, immediately testable changes reduces risk considerably. This can provide exceptionally large boosts to organizational agility.
- Disaster recovery – utilizing automation to quickly provision an application stack in another region is quite a popular DR strategy in the cloud. This cost-effective strategy avoids paying the DR environment until such time as it needed and the rapidly provisioning it via automation in the event of a disaster. While a very efficient way to avoid incurring doubled production costs to provision a full-sized DR environment, this solution only works for the organizations that have embraced automation and have reliable tooling in place for rapid provisioning.

All these benefits combined present a compelling argument for making sure that database specialized personnel embrace DevOps principles, but getting buy-in from those same database personnel is often a challenge. Embarking on projects of sweeping organizational change is always tricky but there are a few effective ways to encourage database specialized personnel to adopt these practices without any heavy-handed measures. House of Brick has participated in many cloud migration projects and can offer these techniques as effective in getting database specialized personnel on board with DevOps techniques:

- Require the use of automatic deployment pipelines coupled with infrastructure-as-code repositories right at the beginning of a cloud migration project. Restrict database personnel to read-only access to the AWS console. This encourages them to quickly embrace the pipelines and scripts as the only means of accomplishing database tasks.
- Insist that database specialists use the same operational tooling as the larger organization. This includes log centralization tools, monitoring and alerting tools, etc. Prevent the DBAs from setting up a closed ecosystem of Oracle-specific tools around the databases.
- Encourage database specialized personnel to get AWS training and certification. If the database specialists understand the cloud architecture and are participants in designing it, they are more likely to utilize it in an effective and collaborative manner.

A DevOps model is not only flexible in support of project/business needs, it can lead to cost savings when applied to pay-as-you-go cloud consumption models. Most organizations that House of Brick assists with the cloud operations end up having wasteful database usage. This is typically due to legacy operational

models for use of non-production databases that have persisted from their on-premises origins. In an on-premises environment, it is extremely common to have several dedicated non-production databases that are reserved for the use of developers or testers.

In addition, there are usually one or more full size databases that are clones of the production environments serving as performance or integration test beds. This approach makes perfect sense within the static confines of an on-premises datacenter environment but makes little sense when persisted in the AWS cloud environment. In an AWS environment, where the complete act of provisioning a brand-new database is measured in minutes instead of weeks, there is little reason to have dedicated non-production databases. These resources are billed by the hour even on days when they are not being used, and there are far too many of those days. Encouraging DBAs to embrace DevOps automation and provide on-demand solutions for provisioning of non-production databases can result in significant cost-savings from non-production environments.

CONCLUSION

An understanding of all the cost traps and risks for running an Oracle database solution is important, but equally important is understanding their relative magnitude. While the exact math will differ depending on the organization, the general rule for achieving cost savings with Oracle database solutions in AWS is to address them in the proper order.

The Oracle Licensing Cost Trap is easily the biggest cost trap and addressing it first should be a priority. Using the House of Brick best-practice techniques to leverage proper rightsizing, open-source engines on the Amazon RDS service, Amazon RDS for Oracle Standard Edition 2 License Included, efficient consolidation and archival, and effective strategies for Oracle license wind down will go a long way towards keeping Oracle licensing costs reasonable in AWS.

After licensing, the traps relating to organizational innovation and agility are the next priority. Embracing the advantages of the Amazon RDS managed service does wonders for addressing the Oracle Change Fear Factor, the Oracle Database Operational Cost Trap, and the Organizational Silo Trap. Adopting DevOps practices will also contribute significantly to addressing the Oracle Database Operational Cost Trap and the Organizational Silo Trap.

House of Brick has a good track record of using this approach to help clients prioritize initiatives, reduce Oracle costs, and trim budgets in AWS without sacrificing functionality or operational rigor. Using the advice from this paper any organization should be able to chart a cost-effective path towards migrating Oracle database workloads into AWS. There are significant cost savings opportunities in migrating to AWS that are beneficial for entire IT organizations and it is important to not let Oracle be the factor that derails taking advantage of those opportunities.

House of Brick strongly recommends that any organization considering a move of Oracle database workloads to AWS consider and apply the principles outlined in this paper. Explore the possibilities and options thoroughly before planning, and review these resources for further information:

- Amazon RDS for Oracle page[6]
- Amazon Aurora[7]
- AWS Database Freedom page[8]
- Amazon Database Migration Accelerator page[9]
- AWS Database Migration Service page[10]

Anyone with specific questions should reach out to their AWS account representative to get support and assistance in learning about those opportunities or reach out to House of Brick for assistance specifically with Oracle in AWS architecture and licensing.

[6]https://aws.amazon.com/rds/oracle/

[7]https://aws.amazon.com/rds/aurora/?aurora-whats-new.sort-by=item.additionalFields.postDateTime&aurora-whats-new.sort-order=desc

[8]https://aws.amazon.com/solutions/databasemigrations/database-freedom/

[9]https://aws.amazon.com/solutions/databasemigrations/database-migration-accelerator/

[10]https://aws.amazon.com/dms/