The HIDDEN COSTS of Not Properly Managing Your Databases

PERCONA

Table of Contents

The Hidden Costs of Not Properly Managing Your Databases

Not properly managing your database environment can kill your business quicker than any competitor or market shift. Poorly tuned and misconfigured systems, which exist in data centers around the globe, waste millions of dollars daily and can lead to performance issues that damage a company's reputation.

This eBook reveals a few sources of unnecessary database costs that may be lurking in your environment. It also examines some common scenarios to demonstrate how improperly managed databases can hurt your bottom line.

Scaling

In this section, we'll look at the costs associated with scaling your environment and steps you can take to reduce those costs. To show how costs can rise as your environment grows, we'll start with an example.

In a small database setup, we have a single primary-replica with a database size of 500GB. Traffic is steady, and this translates into 1000 IOPS on the primary. The database is hosted on Amazon Web Services (AWS). For this example, we're using Amazon Relational Database Service (Amazon RDS) MySQL, a multi-availability zone setup, and a single replica to help with reporting.

A common way to do backups in AWS is to set up Elastic Block Store (EBS) snapshots of the replica. We're assuming that the central processing unit (CPU) is at around 50% of capacity and that about 5% of the data will need to be in memory for optimal performance (in this case, starting at 25GB of hot data in memory).

To support this database setup in RDS, we would need:

- Two instances (one for the primary and one for the replica) db.r6g.xlarge servers (four REAL vcpu graviton, with 32GB of memory) Memory optimized instances are the best suit for MySQL
- primary replica set, multi-AZ for the replica and primary
- 1TB of provisioned input/output operations per second (IOPS) SSD over two volumes (io1)
- 1000 IOPS on the primary and replica (default)
- estimated 7TB of storage for snapshots

Based on these requirements, the AWS calculator estimates a cost of USD $2,847.10 per month, or roughly $34,164 USD a year, in hosting fees. You can also choose other tiers of service or select reserved or spot instances to get different pricing options.

Planning for regular, steady growth

As we move forward, we'll assume the database is growing along with its traffic at about 5% per month. After a year of increasing at this rate, server CPU utilization would be 90%, database size would be 897GB, and we would need 45GB to keep the hot data in memory.

Based on these numbers, we would need to consider upgrading instances as early as the second month due to lack of memory space. Upgrade requirements would happen along the following timeline:

- month 2 - memory
- month 15 - disk
- month 29 - disk and memory
- month 37 - more space
- month 43 - CPU and memory

Many businesses respond to such situations by simply moving up to the next tier of servers. Taking into account additional IOPS, our spend per month would jump dramatically from $2,899 USD to $4,801 from the second month and increase again at 15 months.

Tuning your environment to reduce costs

The costs associated with scaling as demonstrated above are not inevitable. By tuning and auditing the environment, we can arrive at very different results. Using a conservative estimate, we could, for example:

- achieve a 25% boost in performance
- reduce 5% monthly growth to 4%
- shrink the database by 10%

Doing this would allow us to delay upgrades as follows:

- month 5 - memory
- month 21 - disk
- month 39 - disk/memory/CPU

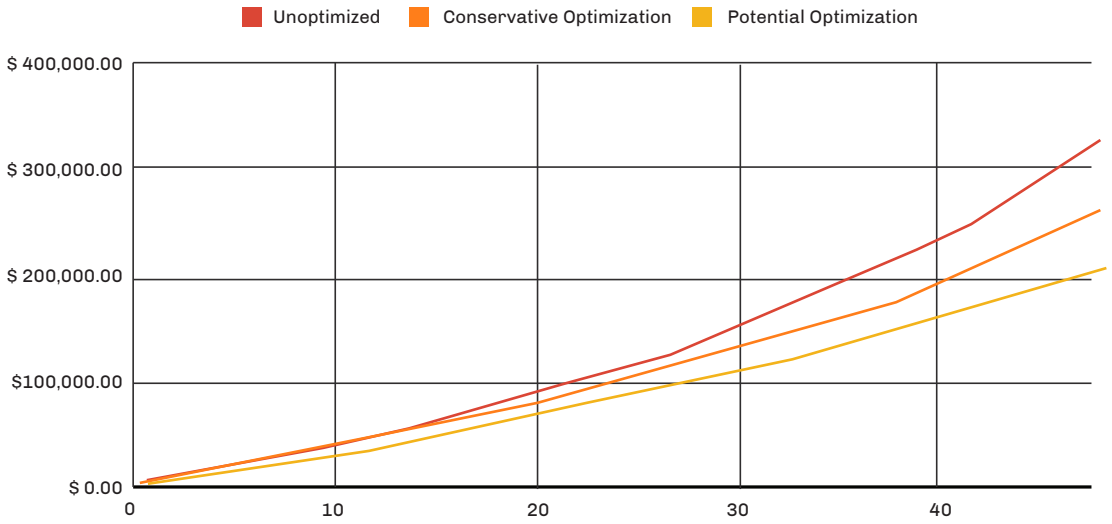
Based on these fairly minor savings, we could postpone upgrading our servers by four months initially and even less frequently in the future. This would save nearly $6,000 USD in the first year, and a significant figure over four years based on smaller data size and performance enhancements.

Through careful analysis and optimization, we could deliver up to a 50% improvement in performance and free up between 20 and 25% in capacity. At this level, the cost savings are considerable, as we would require just two upgrades in three years:

- month 13 - memory
- month 34 - disk

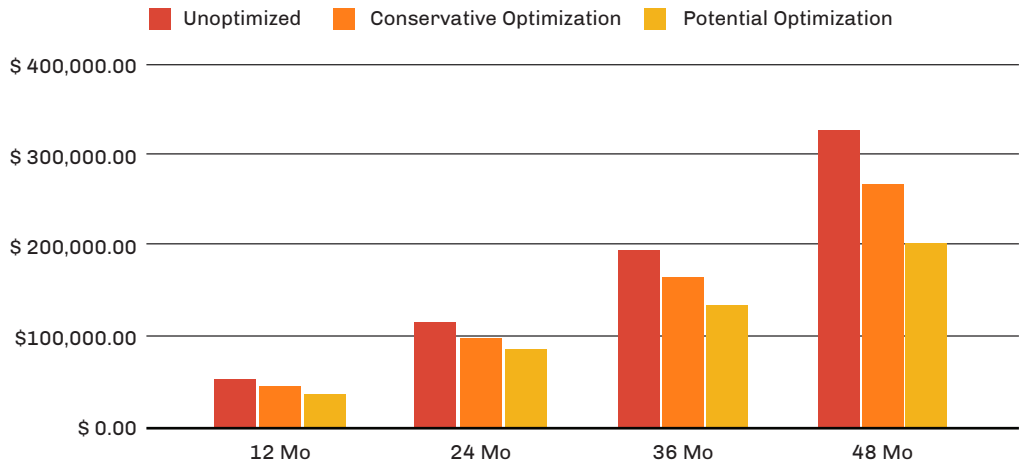
Database Cloud Costs Over Four Years

Database costs for a single application (primary-replica + AZ failover)



Cloud Costs Savings for Optimized Databases

Database costs for a single application (primary-replica + AZ failover)



Optimizing your data can lead to significant savings

In four years with just three servers, **you could save $57,000** (based on a conservative estimate) and **up to $120,000** in total.

Over 10 applications, you could **realize savings of over $1 million**.

Other cost-saving measures

Performance enhancements are not the only way to save costs. Other options may be available depending on your environment.

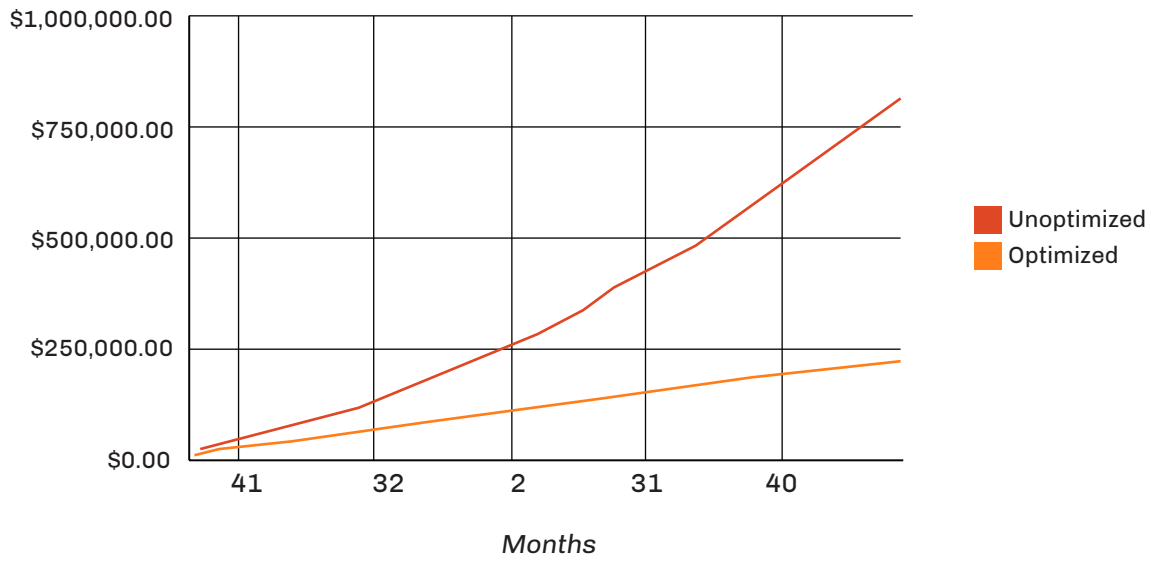
In the example above, we used Amazon RDS. RDS limits backups to EBS snapshots. If you're running your databases outside the RDS environment and onto an Amazon Elastic Compute Cloud 2 (EC2), you can have better control over your backups since you have access to the filesystem. For that particular use case, you can take regular MySQL backups using Percona XtraBackup. Percona XtraBackup is a set of tools that provide a method of performing hot backup of MySQL data while the system is running.

In this EC2 setup, instead of keeping the last seven days of snapshots, we'll keep one backup copy on disk and send the other backups to Amazon Simple Storage Service (Amazon S3). Seven days of snapshots on EBS costs about $16,000 USD per year, while using Percona XtraBackup and Amazon S3 costs about $6,500 per year.

Of course there are always trade-offs for cost savings to consider, such as ease of use and speed to recover. But even storing a single EBS snapshot and then using Percona XtraBackup with Amazon S3 for long-term backup can provide significant savings.

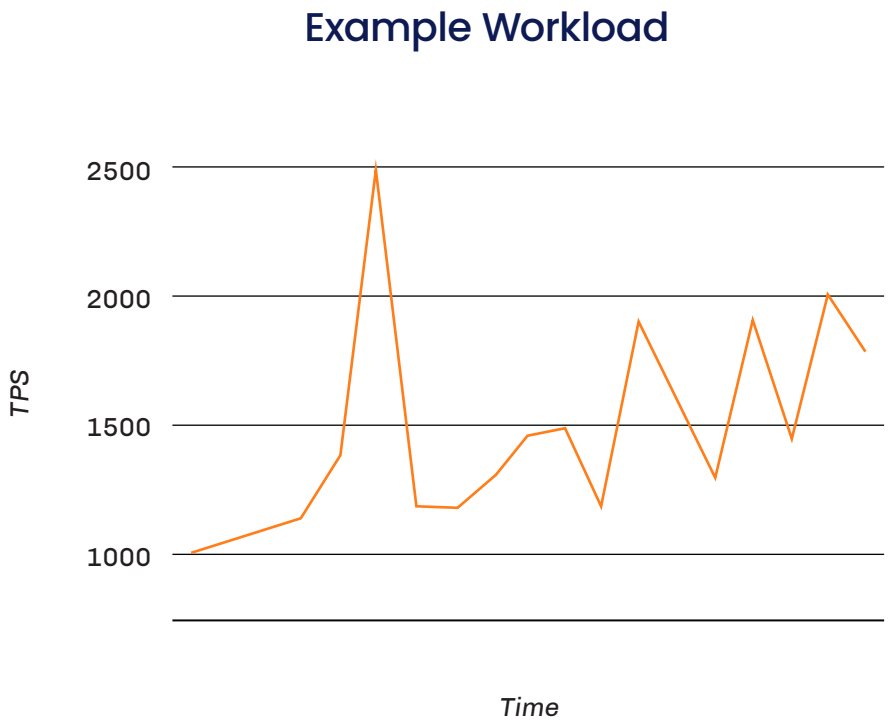
A recent Percona client was able to see a 90% reduction in their read-heavy workload and actually turned off obsolete servers that had previously been required to support their application. Percona helped them cut their direct costs by two-thirds.Their savings over the subsequent two years appear below:

Recent Optimized Customer Example



Database usage spikes

The examples in the prior sections assume a linear growth in terms of application and database usage. This means that, as long as you control the addition of new features as your user base grows, you can predict when you will need hardware. However, each application has its own fingerprint, and in most environments you are more likely to see sudden spikes similar to this:



The first reaction of many businesses to a database usage spike like the one above would be to upgrade their hardware then tune. The cost of the hardware upgrade, however, would sink any potential cost benefits and may not have been needed if tuning had been done first.

Preventing the spike could have saved tens, or even hundreds, of thousands of dollars. Unfortunately, the reason behind a spike may not be easy to find and can result in unnecessarily increasing costs.

Another potential source of performance spikes is your company's application release cycle. Each application uniquely grows, expands, and changes on a regular basis. To stay ahead of problems, every release of new code should go through a rigorous performance review to identify hidden problems and avoid extra costs.

Case study - Database usage spike

A Fortune 500 company encountered a database usage spike. Their databases had been running well with steady but controllable growth for eight months, but in month nine, things quickly went very wrong.

A critical component of their business was certifying professionals through a testing application. During a busy period, these certifications completely stopped, which delayed certifications for thousands of employees and clients for two weeks. When they realized there was a problem, the client contacted Percona to request expert assistance to identify and fix the issue.

The number of users accessing the application remained the same, and page views on the website were steady, but the number of queries to the database skyrocketed. None of the queries hit their threshold of 1 second, which would have flagged them as problem queries.

After extensive analysis, it turned out that one query was taking 250 milliseconds (ms) to execute and had caused the certifications – and the company – to grind to a halt. That solo query was being executed 25,000 times per page when certain conditions were met, and those conditions had not been met until the ninth month after the application was released. This seemingly benign query destroyed the company's ability to service its clients.

Fixing the code was the correct solution. Proper indexing of the tables dropped the query time from 250ms to 50ms. This change enabled the certification process to start back up again and operate until the code could be fixed.

Key takeaways:

- Although your system may appear well-tuned, under the surface, conditions might not be optimal.
- Small things matter. Even queries that seem harmless should be optimized to prevent future issues.
- In the long run, it will cost you less to find experts who can help you proactively identify and remediate problems on an ongoing basis before they cause extensive damage.

Cloud lock-in

Over the past few years, many companies have migrated applications to the public cloud as a way to quickly and cost effectively increase development speed and scale to meet growth. However, moving to the cloud comes with lock-in risks.

Cloud providers charge extremely high egress fees – potentially thousands of dollars per transfer– depending on the pricing tier and where you transfer your data (e.g., across the Internet or AWS Regions, through AWS Availability Zones, to your on-premises data center, etc).

For example, transferring 25TB of data off the AWS platform may cost about $0.085 per GB, or $2,125 per transfer – or even more, depending on where you're sending it (e.g., locally or internationally). In Google cloud, the cost may be about $2,000 per transfer. For enterprises that routinely move a lot of data (e.g., across multi-cloud infrastructures), costs can add up fast.

If you decide to move applications to the public cloud, be very clear on the structure of your architecture and the potential costs you may face if you want to move your data. AWS itself acknowledges the relationship between architectural design and data transfer fees, stating that "data transfer charges are often overlooked while architecting a solution in AWS. Considering data transfer charges while making architectural decisions can help save costs."

Unplanned downtime

Unplanned downtime can inflict very real damage on both your bottom line and reputation. The market is littered with the remnants of companies that did not adequately plan for future scaling or simply missed important issues in their IT infrastructure.

Sources vary, and data depends on a range of variables (industry, seasonality, size, etc.), but most reports estimate that the average cost of unplanned downtime is approximately

$300,000 to $500,000 per hour,

or **$5,000 to $8,000 per minute**.

Although it's impossible to completely eliminate downtime, having a disaster recovery system in place can minimize losses. The most common approaches to disaster recovery are restoring from backup, manual failover to a replica database, and automated failover.

Keep in mind that restoring the database from backup or doing a manual failover may take minutes or hours depending on the severity. Automated failover can minimize downtime, but not all solutions are created equal. Some can take several minutes or several hours to restore proper service.

Whatever you do, it's important to choose a solution that aligns with your business requirements for availability and uptime.

Cloud provider outages

Sometimes an outage is not due to a problem with your database and/or application. Occasionally your cloud provider is the one having technical issues. You should factor this scenario into your disaster recovery planning and consider mitigation strategies, such as adopting a multi-cloud environment.

Slow performance

Customers expect instant access to information and applications, and they have little to no tolerance for slow-loading systems or interruptions.

A Google study found that if your application does not respond in 3 seconds or less, more than half of mobile website users will leave your site. Even if your system is up and running but is slower than usual, impatient users may perceive this as an outage and go elsewhere.

It's hard to calculate the impact on your reputation from these incidents, but the damage to your bottom line is significant. Large social media and gaming websites, for example, can lose tens of thousands of users (or in some cases, hundreds of thousands) due to slow-running applications and websites, showing just how essential it is to optimize your applications and websites to ensure the fastest performance.

According to Amazon, every 100ms of latency costs 1% in sales. If you're a mid-size enterprise (per Gartner's definition) with annual revenues of $50 million, that's a **half-million dollar loss** in sales as a result of a mere 0.001 minutes in downtime.

Poor database security

Old database versions, insecure backend access, misconfigurations, bad passwords, anonymous user policies, and lack of insight into vulnerabilities all are security hazards that can result in dire consequences and high costs.

Over the past several years, database misconfigurations in the cloud have exposed billions of user data records to cybercriminals and cost organizations an average of 4.24 million per incident. (IBM and Ponemon Institute, 2021)

Almost all database security problems are a result of overlooked or missed steps caused by human error. In fact, attacks on misconfigured databases are increasing as hackers use free or cheap tools to breach public databases, usually unintentionally left unsecured by developers. Over the past several years, database misconfigurations have exposed billions of user data records to cybercriminals and cost organizations an average of 4.24 million per incident.

Of course, no one sets out to mismanage their databases, but security is impacted drastically by things like:

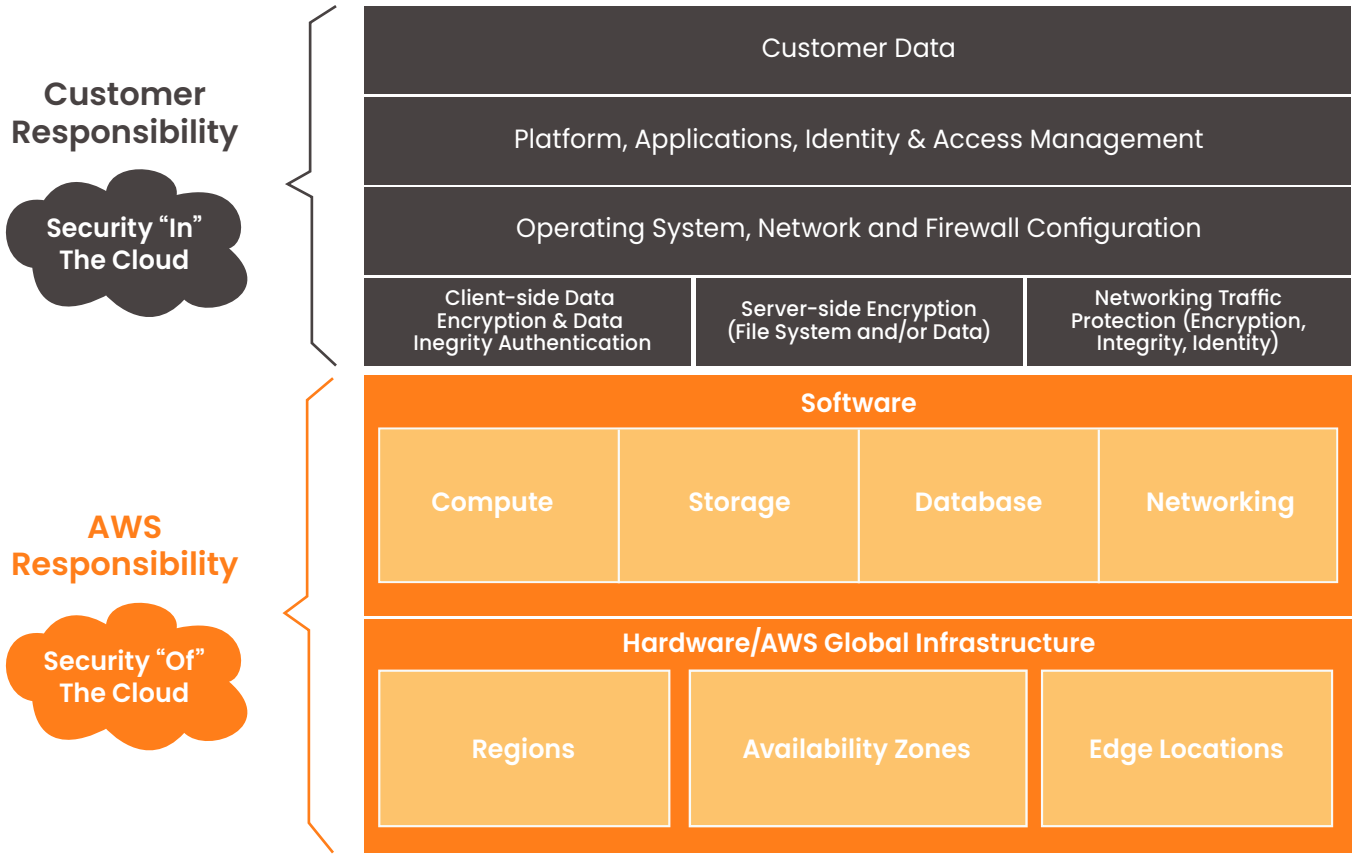
- **Underqualified and undertrained staff being responsible for maintaining the database infrastructure.** An increase in tooling, automation, and push-button deployments are becoming the new normal. These can lead to bad setups that miss the most basic security steps (like a password reset). The industry is moving toward empowering developers, but database setup, security, and performance are not normally top priorities for developers.
- **Staff spread too thin.** Some studies suggest that 80% of companies are running one or more outdated or buggy releases. Companies are now running hundreds or thousands of databases in their data centers. It's easy for a small team to miss a few databases when dealing with them at scale.
- **Not understanding who is responsible for security in the cloud.** The shared responsibility model of security is a critical concept to understand if you run applications in the cloud. Cloud vendors handle the security of the cloud, but you're responsible for the security of your environment in the cloud.
- **Managing massive amounts of data.** Just the sheer amount of data and the volume of data-bases that teams need to manage means the number of security incidents will inevitably increase.

Ensuring robust initial database setup and configurations, keeping software up to date, and prioritizing database maintenance go a long way to protecting your environment from costly security problems.

Cloud shared responsibility security model

Public cloud vendors operate on a shared responsibility security model in which they manage security of the cloud (e.g., identity and access management and network and firewall configurations), and you manage the security in the cloud (e.g., database, compute, and storage).

Here is the AWS definition of the model:



PERCONA

How Percona can help you avoid hidden database costs

We've covered just a few hidden costs that you can incur if you don't properly configure and manage your databases. Reducing these risks requires a high level of database expertise to make sure your environment performs well for your customers.

But not all companies want (or can afford) to maintain a full-time team of DBAs to keep their databases optimized. Percona Managed Services can help by providing database support at a fraction of the cost of a dedicated, full-time DBA.

The Percona team provides tailored managed database services for your business, helping you lower costs and manage your database complexity. And we provide deep operational knowledge of MySQL, PostgreSQL, MongoDB, and MariaDB databases, the cloud platforms of Amazon AWS, Microsoft Azure, and Google Compute Engine, and Amazon's DBaaS offering of RDS and Aurora.

Learn more about how Percona Managed Services can help you.

For more information, please contact us at:
+1-888-316-9775 (USA), **+44 203 608 6727** (Europe), or via email at **sales@percona.com**

170123