Cockroach Labs

The True Cost of a Cloud Database

A framework to help guide your decision

Calculating the costs associated with a database may seem straightforward, but it's not. There are a range of concerns and long-term costs associated with this choice. And today, with the introduction of cloud databases, yet another layer of consideration has been added: cloud provider costs.

To help with this choice, we've pulled together a list of items to consider when thinking through the total cost of ownership of a database. This is not meant to be a definitive guide as every project and every database is different. Rather, we present a framework to help with your decision.

As always, we endeavor to make this as pragmatic and correct as possible. If you feel something is missing/misrepresented, please reach out to the team at Cockroach Labs and let us know.

www.cockroachlabs.com

Costs of a database in the cloud era

We've broken the true database cost into three areas of concern, each with a varying degree of exactness. This document uses this framework to help guide through this consideration.

Hard Cost	Operational Cost	Soft Cost
Costs of servers, instances, technology, and software	Costs associated with managing the database	Risks scenarios and rewards with different databases
Cloud instances	Support	Indemnification
Software licenses	Operations	Competitive risk
IOPS	Scaling/sharding	Vendor lock-in
Storage	Disaster recovery	Attracting talent
Egress	Planned downtime	
Load balancers	Unplanned downtime	
Synchronization	Write bottlenecks	
Data integration	Multi-region	
	Performance	

The hard costs of a cloud database

You can't run software without hardware. Regardless of where your database runs, you will still need to consider hardware or cloud costs and it is important to think through the full slate of items that will need to be considered across the various offerings.

Both AWS database offerings, RDS and Aurora, present a complex set of configurations that each add costs. Spanner simplifies the equation down to instance, storage, network, and backup with multi-region adding costs. CockroachCloud packages all of this into a single price and simplifies the process. It reduces variable cost and makes it more predictable.

	Hardware	Software	Other
OSS PostgreSQL/MySQL	• Spin up instances in a cloud provider • Variable and complex across cloud provider, additional costs for load balancers, security, network, storage, etc	• Download and use OSS software • License VMs and operating system • If running on Kubernetes, there are platform costs	Variable and complex across cloud provider, but options include egress and additional services for security, network, storage, etc
AWS RDS[1] (PostgreSQL or MySQL)	• Instance Types - Choose the size of instance. • Durability (resilience) approximately doubles cost for multi-AZ deployment, which is single region only. • Reserved instances are the premium cost • Add cost for required storage • Backup storage included in storage cost		• IOPS - variable I/O capacity to storage • Egress costs - transfer of data out of instance (free up to 1GB
AWS Aurora[2] (PostgreSQL or MySQL)	• Instance Types - Choose the size of instance and choose between standard or memory-optimized. • Added cost for "global database." If you want to survive failure of a region for reads only, then add cost for write I/O across regions • Reserved instances are the premium cost • Storage costs are rated by usage per GB • Additional cost for backup storage included in storage cost if more than 1 day and across regions • Point in time recovery or "backtrack" is additional cost • Snapshots are an additional cost		• IOPS - variable I/O capacity to storage • Egress costs - transfer of data out of Aurora
Google Spanner[3]	• Instance - GCP offers various sizes of instance, however you should check to see global availability • Storage - Various storage per node is chosen • Network - Ingress is unlimited and free, while charges accrue for egress every month. • Backup - Time-based and charged by size of backup		• Egress costs - transfer of data out of regions • Development costs could grow as each dev will need a cloud instance (no local version)

CockroachDB

Simple pricing includes options for size of instance, IOPs, storage, and load balancing. Backup and restore, point in time recovery and redundancy for resilience all included.[4]

[1] https://aws.amazon.com/rds/postgresql/pricing/
[2] https://aws.amazon.com/rds/aurora/pricing/
[3] https://cloud.google.com/spanner/pricing
[4] https://www.cockroachlabs.com/pricing/

The cost of scale

For most cloud databases, Scale is typically accomplished by increasing the size of the instance. However, with this approach, you are limited with the max size available to you. What happens when you want to scale beyond this, or need global scale?

Some databases, like AWS Aurora, allow you to expand beyond a single instance (RDS) and allow for multiple instances. However, this is for reads only and limits the transaction volume you can handle as there is no capability to scale write nodes. You still face the size limit. Further, this single write node configuration limits your ability to scale across broad geographies as you will always encounter physical latencies for write access.

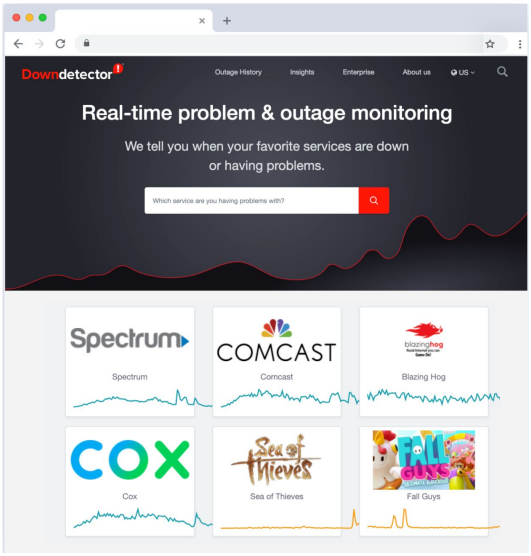
If you choose to scale an open-source database like PostgreSQL or MySQL, you will eventually need to shard the database. There are massive costs associated with this approach. For one, you will need to modify your application, which introduces risk. You will also need to configure a new instance and cut over to this new configuration at some point, typically in the middle of the night. You will have extra costs associated with the hardware and the pain it causes for the team… and this is all the best case scenario. If something goes wrong in this process, you wind up with unplanned downtime. Also, management costs of a sharded database increase exponentially with each new shard.

CockroachDB allows you to scale easily by simply adding additional nodes to a cluster, and the database will take care of creating ranges (similar to shards) that are then balanced across the nodes according to policy set at the table level. There is no need to perform costly, painful, manual sharding.

With CockroachDB, every node is a consistent gateway to the entirety of the database, which means every node can service reads and writes and execute queries across the distributed cluster. This eliminates the write bottleneck concern and ensures scale for not only size of database but also for volume of transactions.

Disaster Recovery, Survival and the Cost of Downtime

Everything fails. Everything. The cost of a temporary or catastrophic challenge needs to be considered when choosing the database to deploy your application. While there are many reasons an application might fail, the database is a main cause of many outages. Old versions, write bottlenecks, memory issues, locked transactions, misconfigurations, hardware failure. You need to prepare for these inevitabilities, because they will happen at some point.



There are several sites that track downtime across many popular sites. Downdetector.com (left) provides a running list of downtime reports and a graph over time so you can identify peaks and server issues. For many of these events, you can also investigate the comments, which will outline what happened.

Many organizations also offer complete insight and transparency into their own issues and will speak about downtime and the challenges on their engineering blog.

Planned downtime also needs to be considered when choosing a database. Often you will need to upgrade the database software or make an architectural change. You may even want to modify schema. With many databases this involves a planned outage (often in the middle of the night) that will impact business and wear the patience of the team that has to execute the change. These moments are risky too, as anything could go wrong and a short planned outage could turn into a larger issue. They simply aren't fun and are to be avoided.

The business impact of planned and unplanned downtime

Both planned and unplanned downtime can result in significant cost and while this spend may not be easily calculated, it should still be considered as part of your database choice. Each business is different and the impact of downtime is different for each, but these are a few items to consider when calculating the cost of downtime for your business:

- Loss of revenue - Missed opportunity to conduct online commerce or engage a prospect
- Reputation impact - Consumers may just go on to the next offer, your competitor
- Client satisfaction - Loss of trust in your product or service due to observed issue
- Regulatory costs - Jurisdictional regulations sometimes fine organizations for data issues
- Legal liability - In extreme cases, lawsuits may be filed associated with data loss

There are some pretty simple and more straightforward technical reasons why data loss and downtime can be an issue. Often we find ourselves dealing with an extended RPO/RTO period and will endure the following technical costs:

IT costs associated with downtime

Root Cause	Identification of what went wrong involves an investigation and in many organizations, documentation of the event. Often this takes time and many resources to identify.
Recovery	When a database fails and a backup comes online there is a delta between the two systems and remediation of this difference could take hours if not days to understand as you need to investigate and manually resolve every row.
Equipment	The next section speaks to architectural choices you can take to avoid these downtime events, but typically it will involve extra, or a passive, backup database.
Productivity	Downtime not only affects the team that has to deal with the issue, but it can also have an adverse effect on your development teams as they are often called in to help remediate and the downtime stalls deployments and progress.

A costly, active-passive "solution" for survival

As noted, everything fails, so avoiding these issues and additional costs may seem difficult. Many of us have deployed redundant systems and used things like an active/passive setup when we configure a database. This configuration not only adds complexity to the data architecture, it can result in significant extra cost as you now need another database license, hardware, load balancers, etc… for the *backup* system. Not to mention, depending on what you want to survive, you may incur massive costs with egress and data transfer. If you want to survive a regional failure, you will be syncing data across egress bounds of your cloud provider. Finally, even with an active-passive configuration you will still have downtime and additional risk.

CockroachDB

CockroachDB allows you to survive almost any failure. It is a distributed database that stores multiple copies of data so that if a node goes down, every row in a table will still be available. This allows you to avoid the database not being available during planned/unplanned downtime.

The database also allows you to execute rolling upgrades of the database software so you can avoid this downtime as well. You simply restart nodes in the cluster with the new version of the software and the natural survivability of the database will rebalance and resolve the temporary loss of the node during the upgrade.

Finally, with every node active and serving as a gateway to the entirety of the database, CockroachDB eliminates the need for costly active-passive configurations, because every node is active. The database also provides powerful distributed backup/restore capabilities so you can always revert back to a point in time of the database.

While it is difficult to place an explicit number on the cost of planned and unplanned downtime, with CockroachDB you can control the risk and avoid many of these costs.

The cost of performance

Performance and databases has always been a tricky concept. Many database vendors will publish benchmarks that position them favorable against others. However, very rarely are they an apples to apples comparison and many times these differences come down to hardware. This is why the TPC has come up with a way to validate cost per tpmc, which places a dollar amount on transactions per minute for a standard test.

When evaluating the cost of one database over another it is best practice to use a standard benchmark and never trust a vendor-driven benchmark. Further, the cost per transaction, while not perfect, will give you a sense of cost over performance. Typically, if you want an increase in performance you have to increase the size of the machine that is running the database.

Another concern when it comes to transactional cost and performance is the throughput and the upper limit of the number of transactions the database/hardware combination can handle without toppling over. You don't want to incur downtime, so it will be important to evaluate or at least consider this point.

CockroachDB

Gaining performance metrics for a database is often difficult. At Cockroach Labs we endeavor to be as transparent as possible with our numbers and to always publish[5] our current benchmark performance results. We publish our tests so you can execute themselves, and we even include them in our binary[6]. For more information check out our docs and spin up cockroach demo.

We have also run TPC-C at 5K, 10K and even 100K warehouses and we have published our cost per tpmc values so you can try to estimate these costs based on our calculations. However, these are estimates and we always recommend obtaining these numbers on your particular hardware and environment for yourself.

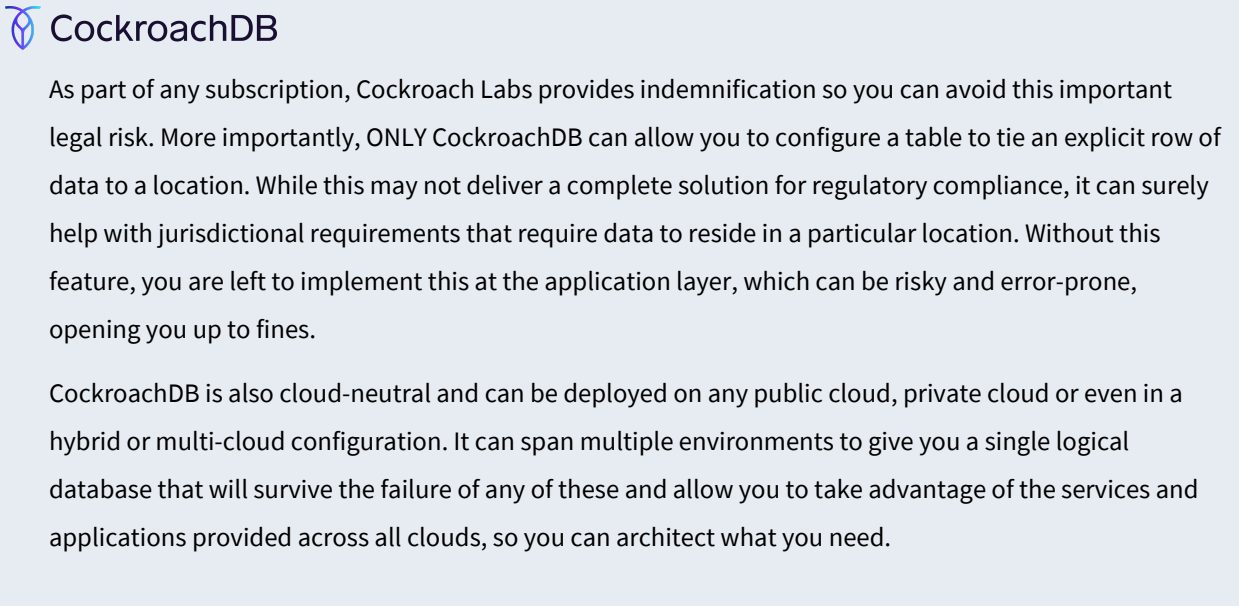
Real risk and some soft costs

While not as easy to measure as some of the other costs outlined above, you may want to consider the impact of risk associated with your database choice. The biggest reason companies choose to purchase a database rather than go with the open source version is to acquire *indemnification*[7]. Indemnification is protection against the database software causing harm to another, and your liability in that scenario. This clause is usually very important with open source software as the license is free, but it comes with risk. That's why many large organizations require indemnification before they allow the software to be used. So, what is the cost of not having indemnification? It really depends on your business and the extent of risk that may be placed upon you.

The second and very real risk is related to regulatory compliance. There are privacy laws being drafted and ratified around the world that set forth guidelines for privacy and protection of individuals' data. This may require data to physically reside in its jurisdictional origin. The web of regulations is incredibly complex and some of them come with significant and very real financial risks. In 2019, Google was hit with a GDPR compliance violation in France that cost them $57M dollars[8].

[5] https://www.cockroachlabs.com/docs/v20.1/performance.html
[6] https://www.cockroachlabs.com/docs/v20.1/cockroach-workload
[7] https://www.scl.org/articles/3030-indemnity-and-limitation-of-liability-provisions-in-software-product-licensing-contracts
[8] https://techcrunch.com/2020/06/19/french-court-slaps-down-googles-appeal-against-57m-gdpr-fine/

Finally, we live in a competitive environment and our data and applications have become increasingly important differentiators. It seems each year we see a market or an apparent incumbent being toppled because of a technological advantage. Unfortunately, sometimes our apps and services don't age as fast as the rate of technological advance. Keeping up with this requires you to be nimble enough to be able to choose and adopt technology that suits your needs. Aligning with a single cloud provider or being tied into their service could present risk and essentially cost as you are tied to their speed of innovation and the services they introduce and make available. A cloud-neutral database will help you mitigate this risk by allowing you to migrate if necessary from one cloud provider to another.



CockroachCloud: Cut costs and get to business quick

CockroachDB delivers on the promise of the cloud. It allows developers to build their next breakthrough application without worrying about complex setup for disaster recovery or without concern of elastic scale. It allows you to tie data to a location and can guarantee transactional consistency so you can be confident in it for even your most stringent system of record workloads. That said, it provides familiar, application and developer-friendly SQL, so you can get up and coding even on the most mundane, general applications.

You can get started today by deploying a managed cluster for free.