Como escalar MySQL?

Carlos Tutte, Senior consultant

24 de octubre, 2023

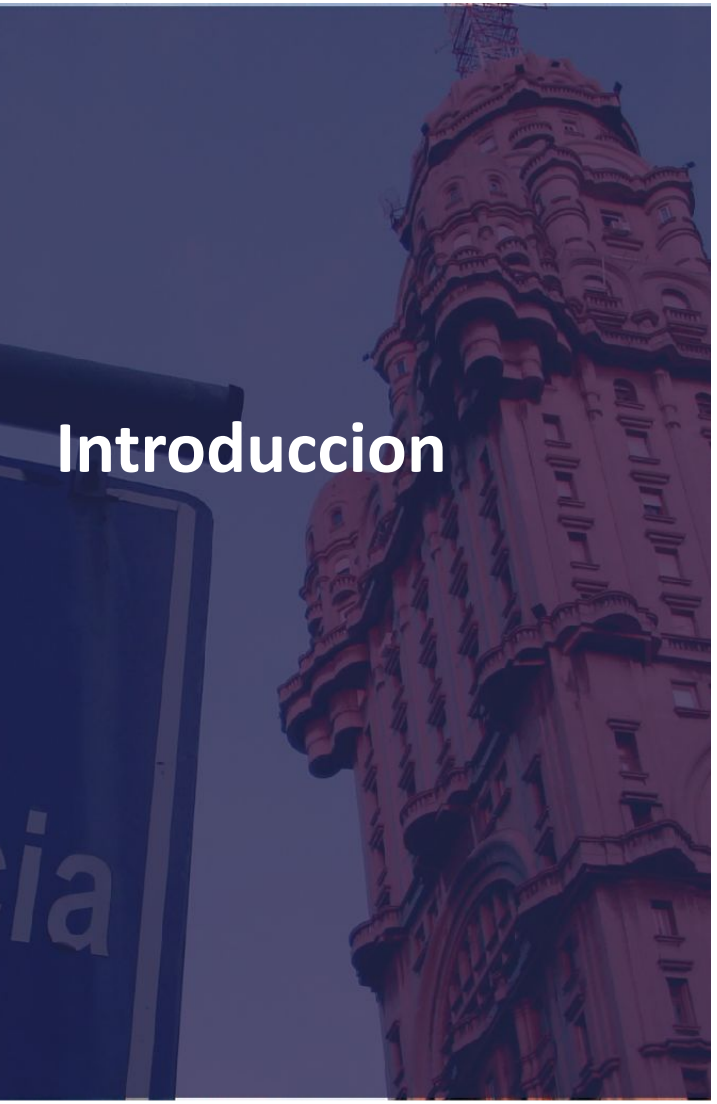
Montevideo, Uruguay 2023

PERCONA UNIVERSITY

PERCONA

Agenda

- Introduccion
- Cuando escalar y porque escalar?
- Escalamiento vertical
- Escalamiento horizontal
  - Replicacion Async/PXC/GR
  - Balanceo de carga
- Queries mas inteligentes
- Caching
- Particionado funcional
- Sharding
- K8s?
- Conclusiones
- Preguntas?

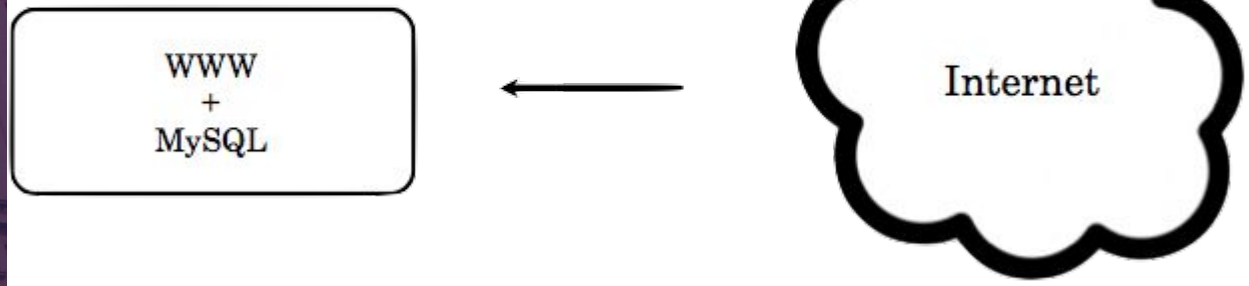


Introduccion

- En esta charla vamos a ver distintas estrategias de como escalar una topología de MySQL para poder servir más clientes y/o ejecutar más queries por segundo (QPS)

Disclaimer: No voy a hacer una introducción formal a lo que es performance/throughput/escalar

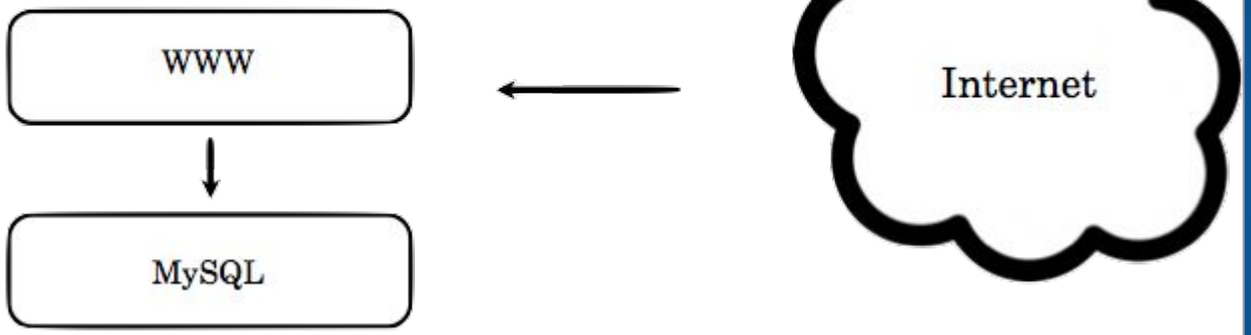
En el inicio



Montevideo, Uruguay 2023

PERCONA UNIVERSITY

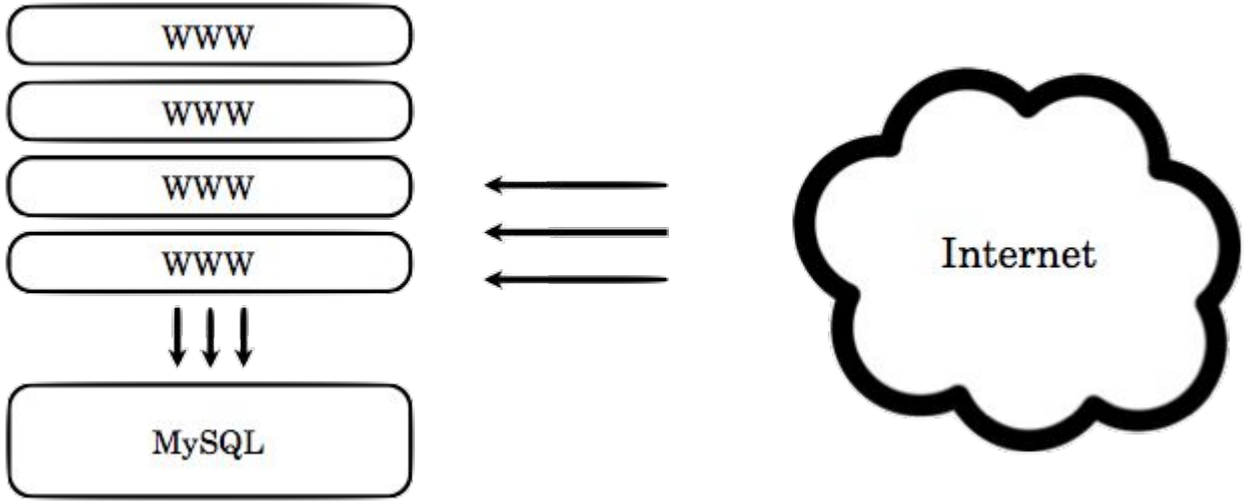
Primeros pasos



Montevideo, Uruguay 2023

PERCONA UNIVERSITY

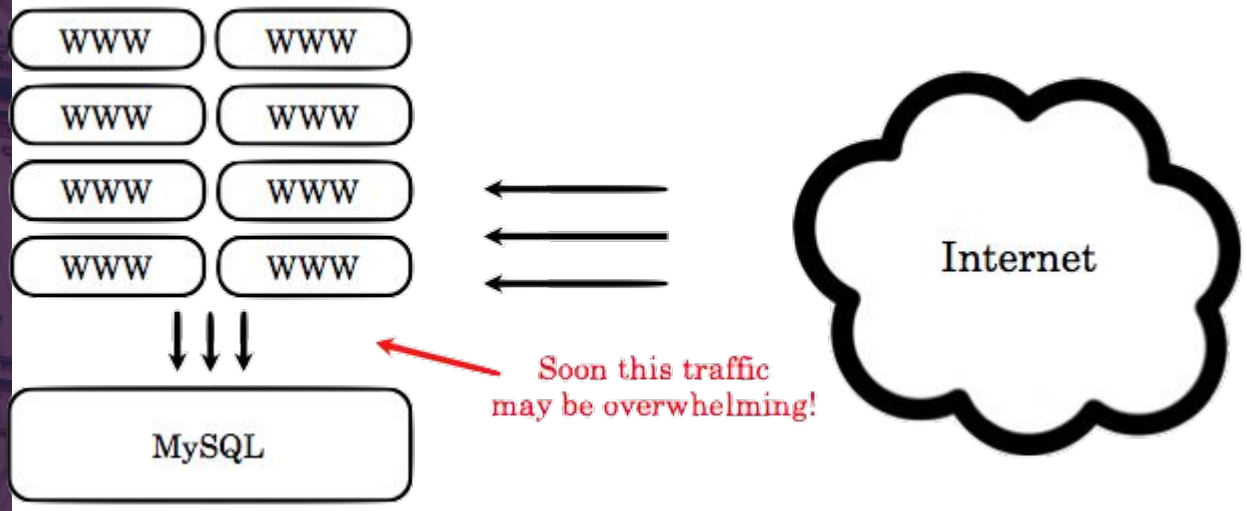
Crecimiento sin problemas



Montevideo, Uruguay 2023

PERCONA UNIVERSITY

Hasta que ...



Montevideo, Uruguay 2023

PERCONA UNIVERSITY

Cuando escalar y porque escalar?

- Cuando?
  - Cuando empiezan los problemas de performance o proactivamente por capacity planning

- Porque?
  - Para adaptarse a las crecientes necesidades de tráfico y operativa:
    - Ganar en redundancia
    - Mejorar throughput/performance/response time
    - Mejorar operativa
      - Tomar backups de una replica
      - Rolling upgrades

Escalamiento vertical

- Es incrementar los recursos del servidor (sin agregar nuevas instancias a la topología)

- Ejemplo:
  - db.r5.4xlarge: 8 vCPU, 16 GB ram

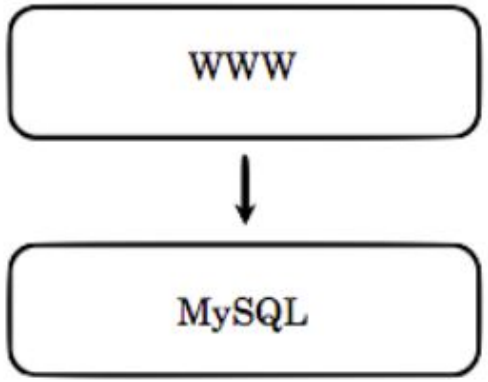
    vs

  - db.r5.8xlarge: 16 vCPU, 32 GB ram

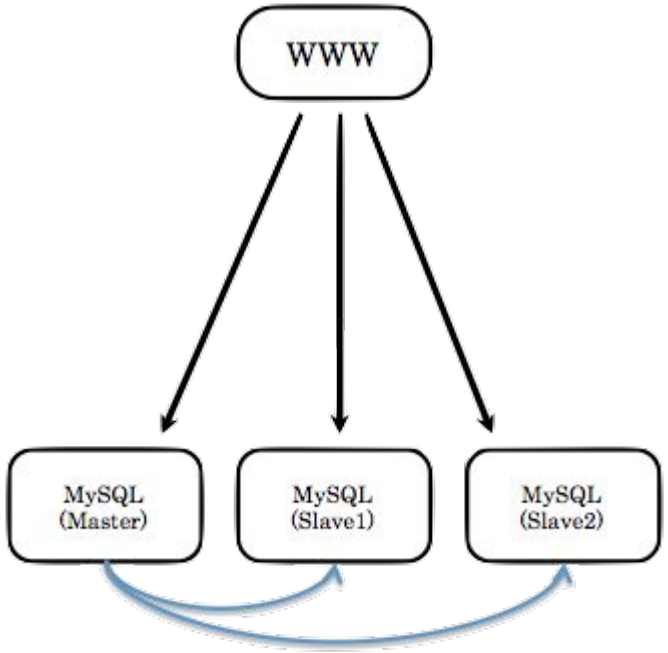
- Facil en Cloud, no tanto on premise

Escalamiento horizontal

- Al escalar horizontalmente se agregan más nodos a la topología



VS



Replicacion

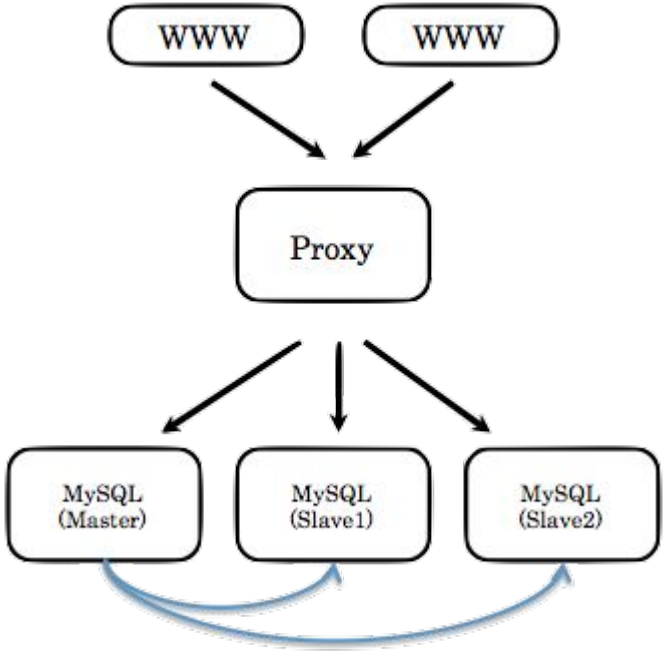
- Distintas tecnologias
  - Replicación asincrónica
  - Percona XtraDB cluster (PXC)
  - Group replication (GR)

- La aplicación debe conocer la topología para poder hacer uso de las réplicas;
  O poner un balanceador de carga

Async: https://dev.mysql.com/doc/refman/8.0/en/replication.html
PXC: https://docs.percona.com/percona-xtradb-cluster/8.0/index.html
GR: https://dev.mysql.com/doc/refman/8.0/en/group-replication-summary.html

Balanceo de carga

- Con un balanceador de carga la aplicación no precisa conocer la topología
  - Puede tener problemas como read-after-write consistency
- Balanceador capa 4: HAProxy
- Balanceador capa 7: Proxysql



PERCONA

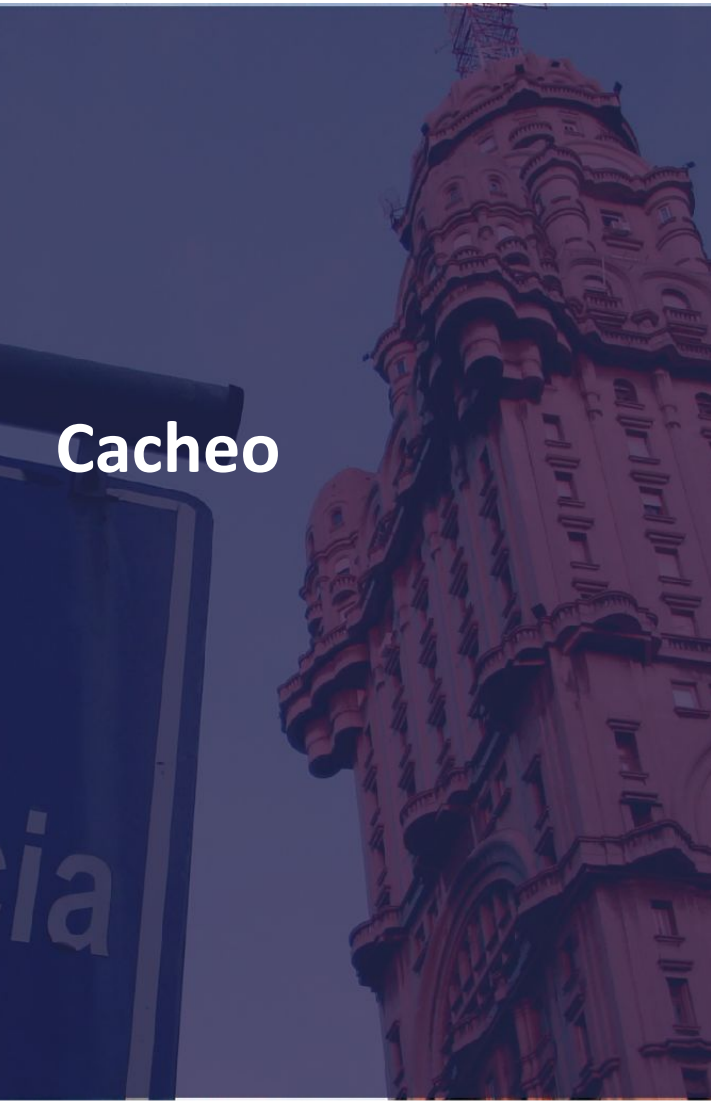
ProxySQL

- Features
  - Load balancing
  - Connection multiplexing
  - Clustering (SPOF)
  - Query rules
  - Query rewrite
  - Query caching (TTL)
  - Query mirroring
  - Query annotation

https://docs.percona.com/proxysql/index.html

Queries mas inteligentes

- Es necesario hacer “SELECT *” ? Se precisan todas las columnas?
- Aplicar filtros en la condición WHERE
- Se precisan todas las rows resultantes o se puede utilizar LIMIT ?
- Ejecutar 100 queries o una única query que haga un range scan?
- Se precisa el resultado exacto o se puede utilizar un estimado?
  - Ejemplo, cantidad de vistas para cada video en youtube



- Guardar el resultado por si la misma query es ejecutada nuevamente
  - Implementar esto no es gratis ni fácil;
    - Memcache
    - Proxysql
    - Query_cache de MySQL (Removed in 8.0)
  - La “rule of thumb” es que el hit rate tiene q ser lo suficientemente bueno como para justificar 3 round trip en un cache miss

- **Problemas!!!!**
  - cache invalidation
  - cache warmup
  - cache stampede

Cache stampede

- Algunas soluciones
  - Lockear la BD mientras se regenera el cache
  - Usar “probabilidad” para regenerar antes que expire
  - Regenerar desde cron jobs

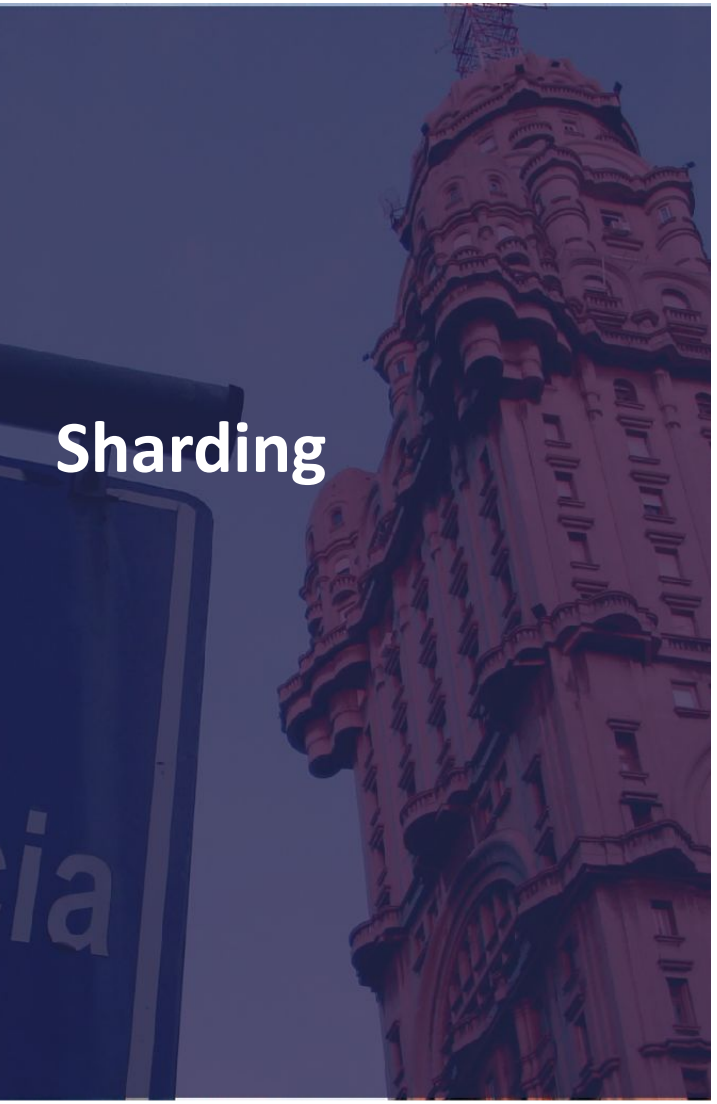
```
function get_cars_sold_count($recurse = 0)
{
    if ($recurse > 2) { return 0; } // Bail out
    if (($cached_object = get_cache('CARS_SOLD_COUNT')) !== FALSE)
    {
        return $cached_object;
    }
    else
    {
        // A mutex is a lock with mutual exclusion - if we have it,
        // everyone else must wait for us to release it.
        if (acquire_mutex_or_wait('CARS_SOLD_COUNT'))
        {
            // We've acquired the mutex - we're in charge of
            // regenerating the cache key and updating it in memcached.
            $query = $db->query('SELECT COUNT(*) AS c FROM Sales');
            $row = $query->fetch();
            set_cache('CARS_SOLD_COUNT', $row['c']);
            release_mutex('CARS_SOLD_COUNT');
            return $new_value;
        }
        else
        {
            // Someone else managed to grab the mutex. It should be safe
            // to repeat this call and retrieve from the cache again.
            return get_cars_sold_count($recurse++);
        }
```

```
// The POW_MULTIPLIER is a trick to help make the probability of regenerating
// exponentially less likely when there is a lot of time remaining.
// For example: A POW_MULTIPLIER of 2:
//  10 seconds out, there is a 1:100 chance.
//  60 seconds out, there is a 1:3600 chance.
//  10 minutes out, there is a 1:360000 chance.
define('POW_MULTIPLIER', 2);
define('MAX_REGENERATION_WINDOW', 600); // 10 mins
function get_cars_sold_count()
{
    if (($cached_object = get_cache('CARS_SOLD_COUNT')) !== FALSE)
    {
        list($expiry, $object) = $cached_object;
        $seconds_till_expiry   = $expiry - time();
        $probability = floor(pow($seconds_till_expiry, POW_MULTIPLIER));
        if (($seconds_till_expiry < MAX_REGENERATION_WINDOW)
            && rand(1, $probability) == 1) {
            if (acquire_mutex('CARS_SOLD_COUNT'))
            {
                // An alternative is not to regenerate now, but write to a
                // message queue that this item should be regenerated.
                schedule_build_cars_sold_count_cache();
            }
            else
            {
                /* Someone else has already scheduled the cache to regenerate.  Continue. */
            }
        }
        return $object;
    }
    else
    {
        return build_cars_sold_count(); // never want this cache miss!
```

Particionado funcional

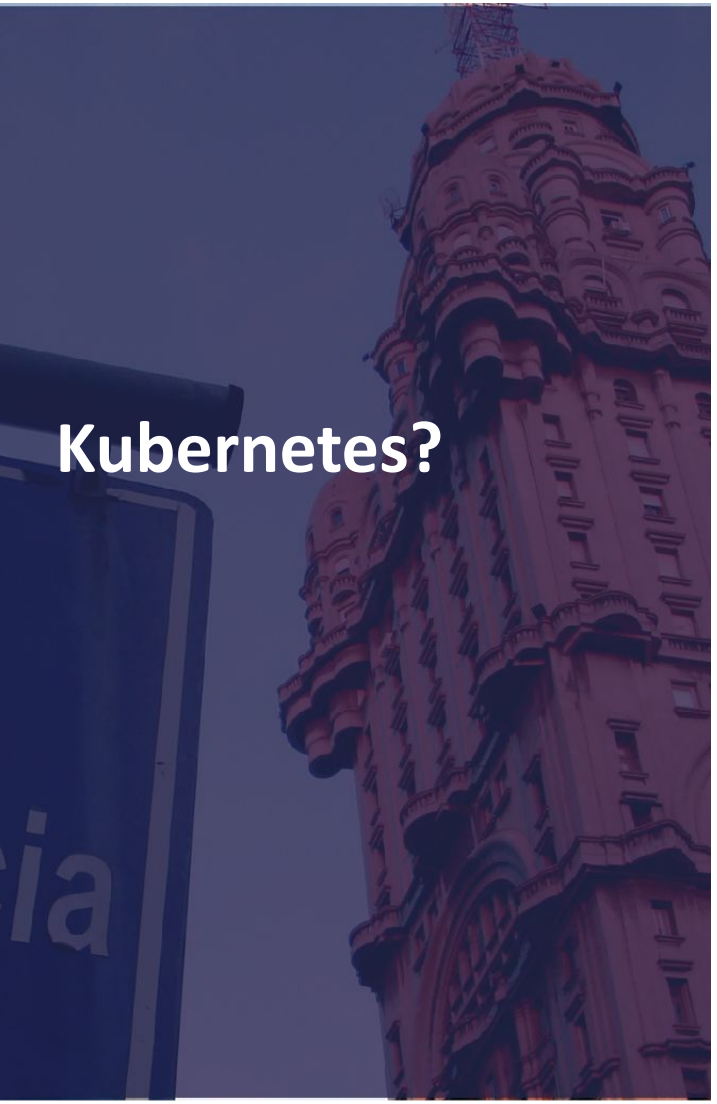
- Identificar tablas/schemas que puedan ser aislados y moverlos a una nueva instalación
  - Ejemplo; si tenemos dos schemas independientes {A, B} en un MySQL tal vez podamos mover cada schema a un nuevo servidor

- Nota: Se debe mover la tabla/schema enteros, sino no es particionado funcional sino sharding



- Dividir una tabla en pedazos entre distintos servidores
  - No hay manera fácil ni directa de hacer esto. Es un trabajo manual
  - https://www.percona.com/blog/mysql-sharding-with-proxysql/
  - La estrategia de sharding es vital para que la carga esté lo más equi distribuida posible

- **Problemas!!!**
  - Dificulta agregacion
  - Dificulta joins
  - Dificulta operativa
    - Upgradear version/modificar schema



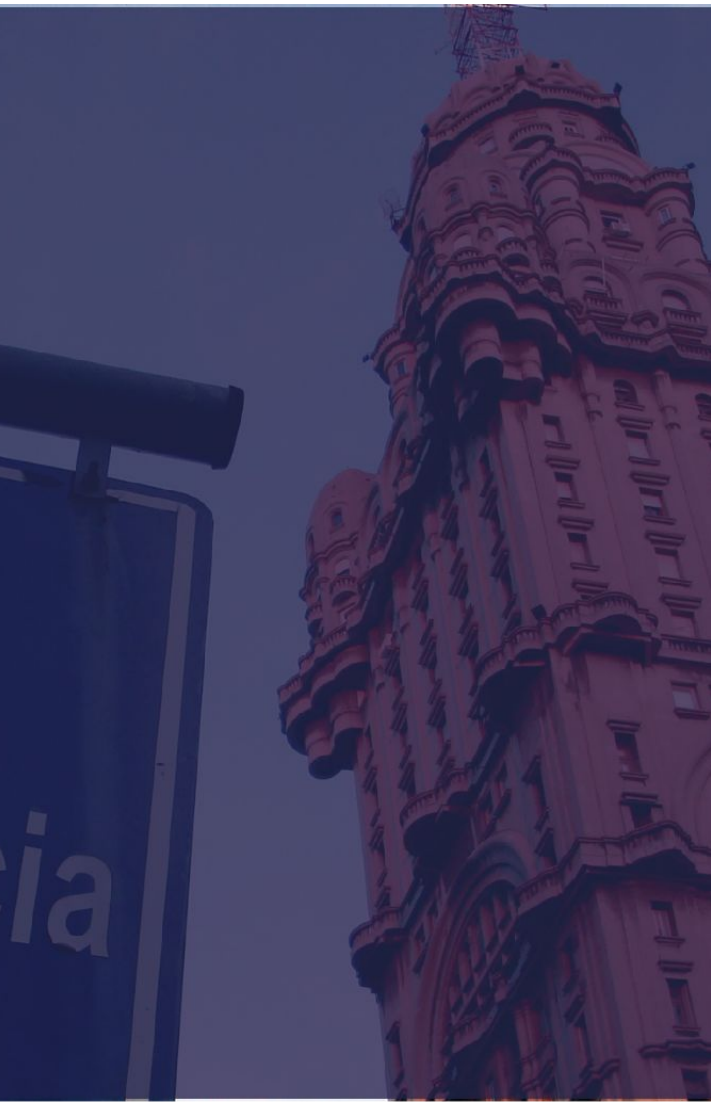
- Permite escalar la operativa
  - Es posible hacer muchos deploys y reconstruir los nodos de una manera rápida y eficiente
- No es una estrategia para usar con cualquier deploy. Sirve para
  - Instalaciones pequeñas
  - No criticas
  - Micro servicios
  - Stateless
- Mas info: https://www.percona.com/software/percona-kubernetes-operators

Preguntas?

PERCONA

Thank you!

PERCONA



Montevideo, Uruguay 2023

PERCONA UNIVERSITY

PERCONA