# Proyecto 1 - Bases II

# Profesor:

Rodrigo Núñez

# Estudiantes:

David Hernández Calvo: 2020041924

Kevin Quesada Jiménez: 2020096838

Esteban Durán Vargas: 2020388144

Andrey Fabián Picado Arias: 2020135773

II Semestre 2021

# Comandos containers de Kevin

docker network create --driver bridge --subnet 12.0.0.32/28 Proyecto1\_Bd2

docker volume create vol\_CFG\_2

docker volume create vol\_shard3\_2

docker volume create vol\_shard2\_1

docker run -d --net Proyecto1\_Bd2 --ip 12.0.0.34 -v vol\_CFG\_2:/data/configdb -p 27017:27017 --name cfg\_server2 mongo mongod --port 27017 --configsvr --bind\_ip\_all --replSet "repcfg" --dbpath /data/configdb

docker run -d --net Proyecto1\_Bd2 --ip 12.0.0.36 -v vol\_shard3\_2:/data/db -p27018:27017 --name shard3\_2 mongo mongod --port 27017 --shardsvr --bind\_ip\_all --replSet "repshard3" --dbpath /data/configdb

docker run -d --net Proyecto1\_Bd2 --ip 12.0.0.35 -v vol\_shard2\_1:/data/db -p27020:27017 --name shard2\_1 mongo mongod --port 27017 --shardsvr --bind\_ip\_all --replSet "repshard2" --dbpath /data/configdb

docker run -d --net Proyecto1\_Bd2 --ip 10.0.0.37 -p27019:27017 --name Arbitro\_Shard\_2 mongo mongod --port 27017 --replSet repshard2

# Comandos containers de David

// docker network create --driver bridge --subnet 11.0.0.0/28 Proyecto1\_Bd2

docker volume create --name vol\_proyecto1

docker volume create vol\_shard1\_1

docker volume create vol\_shard2\_2

docker run -d --net Proyecto1\_Bd2 -v vol\_shard1\_1:/data/configdb -p 27018:27017 --name shard1\_1 mongo mongod --port 27017 --shardsvr --bind\_ip\_all --replSet "repShard1" --dbpath /data/configdb

docker run -d --net Proyecto1\_Bd2 -v vol\_shard2\_2:/data/db -p 27019:27017 --name shard2\_2 mongo mongod --port 27017 --shardsvr --bind\_ip\_all --replSet "repshard2" --dbpath /data/configdb

docker run -d --net Proyecto1\_Bd2 --ip 11.0.0.4 -p 27017:27017 --name Arbitro\_Shard\_1 mongo mongod --port 27017 --bind\_ip\_all --replSet repShard1

// monolitico creado

docker run --name mongolitico -d -v vol\_proyecto1 -p 27016:27017 mongo

# Comandos containers de Esteban

docker network create --driver bridge --subnet 12.0.0.0/28 Proyecto1\_Bd2

docker volume create vol\_cfg\_server1

docker volume create vol\_shard1\_2

docker run -d --net Proyecto1\_Bd2 --ip 12.0.0.3 -v vol\_cfg\_server1:/data/configdb -p 27018:27017 --name cfg\_server1 mongo mongod --port 27017 --configsvr --bind\_ip\_all --replSet "repcfg" --dbpath /data/configdb

docker run -d --net Proyecto1\_Bd2 --ip 12.0.0.4 -p 27019:27017 --name Arbitro\_Shard\_3 mongo mongod --port 27017 --bind\_ip\_all --replSet repshard3

docker run -d --net Proyecto1\_Bd2 --ip 12.0.0.5 -v vol\_shard1\_2:/data/configdb -p 27020:27017 --name shard1\_2 mongo mongod --port 27017 --shardsvr --bind\_ip\_all --replSet "repShard1" --dbpath /data/configdb

docker run -d -p 27017:27017 --name router1 mongo mongos --port 27017 --configdb repcfg/25.5.99.67:27018,25.5.150.109:27017,25.6.191.202:27018 --bind\_ip\_all

# Comandos containers de Andrey

//comandos de la network

docker network create --driver bridge --subnet 172.16.0.0/28 Proyecto1\_Bd2

//CFG 3

docker volume create vol\_cfg\_server3

docker run -d --net Proyecto1\_Bd2 --ip 172.16.0.3 -v vol\_cfg\_server3:/data/configdb -p 27018:27017 --name cfg\_server3 mongo mongod --port 27017 --configsvr --bind\_ip\_all --replSet "repcfg" --dbpath /data/configdb

//shard 3\_1

docker volume create vol\_shard3\_1

docker run -d --net Proyecto1\_Bd2 --ip 172.16.0.2 -v vol\_shard3\_1:/data/db -p 27019:27017 --name shard3\_1 mongo mongod --port 27017 --shardsvr --bind\_ip\_all --replSet "repshard3" --dbpath /data/configdb

//router 2

docker run -d --net Proyecto1\_Bd2 --ip 172.16.0.4 -p 27017:27017 --name router2 mongo mongos --port 27017 --configdb repcfg/25.5.99.67:27018,25.5.150.109:27017,25.6.191.202:27018 --bind\_ip\_all

# Comandos pruebas del cluster funcional

Teniendo el cluster encendido al 100%. Hacemos find y funciona correctamente.

Probando con los shardTags de la siguiente manera dentro del shard 2.1 con el comando docker exec -it shard2\_1 mongo y dentro del mongo.

> use Airbnb

> db.Listings.find({property\_type: "Entire apartment"}).pretty()

y retorna vacío ya que los datos no pertenecen a este shard

Si se apagan aunque sea un shard por completo y se hacen los comandos:

> use Airbnb

> db.Listings.find().count()

en un router, genera un error al hacer la consulta.

Si se vuelve a encender aunque sea una parte del shard, este vuelve a funcionar con normalidad.

y si volvemos a realizar un find dentro del mismo shard, vemos los resultados que sí deben aparecer como sigue:

db.Listings.find({property\_type: "Entire house"}).pretty()

# Comandos réplicas

rs.initiate(

{

\_id: "repcfg",

configsvr: true,

members: [

{ \_id : 0, host : "25.5.99.67:27018" },

{ \_id : 1, host : "25.5.150.109:27017" },

{ \_id : 2, host : "25.6.191.202:27018" }

]

}

)

rs.initiate(

{

\_id: "repshard3",

members: [

{ \_id : 0, host : "25.6.191.202:27019" },

{ \_id : 1, host : "25.5.150.109:27018" }

]

}

);

rs.addArb("25.5.99.67:27019");

rs.initiate(

{

\_id: "repshard2",

members: [

{ \_id : 0, host : "25.6.86.167:27019" },

{ \_id : 1, host : "25.5.150.109:27020" }

]

}

);

rs.addArb("25.5.150.109:27019");

rs.initiate(

{

\_id: "repShard1",

members: [

{ \_id : 0, host : "25.6.86.167:27018" },

{ \_id : 1, host : "25.5.99.67:27020" }

]

}

);

rs.addArb("25.6.86.167:27017")

# Comandos router

sh.addShard("repShard1/25.5.99.67:27020");

sh.addShard("repshard2/25.5.150.109:27020");

sh.addShard("repshard3/25.6.191.202:27019");

sh.addShardTag("repShard1", "Private room in house");

sh.addShardTag("repshard2", "Entire house");

sh.addShardTag("repshard2", "Private room in apartment");

sh.addShardTag("repshard3", "Entire condominium");

sh.addShardTag("repshard3", "Entire place");

sh.addShardTag("repShard1", "Entire apartment");

sh.addTagRange( "Airbnb.Listings",

{property\_type: "Private room in house"},

{property\_type: "Private room in house999"},

"Private room in house"

)

sh.addTagRange( "Airbnb.Listings",

{property\_type: "Entire apartment"},

{property\_type: "Entire apartment999"},

"Entire apartment"

)

sh.addTagRange( "Airbnb.Listings",

{property\_type: "Entire house"},

{property\_type: "Entire house999"},

"Entire house"

)

sh.addTagRange( "Airbnb.Listings",

{property\_type: "Private room in apartment"},

{property\_type: "Private room in apartment999"},

"Private room in apartment"

)

sh.addTagRange( "Airbnb.Listings",

{property\_type: "Entire condominium"},

{property\_type: "Entire condominium999"},

"Entire condominium"

)

sh.addTagRange( "Airbnb.Listings",

{property\_type: "Entire place"},

{property\_type: "Entire place999"},

"Entire place"

)

sh.enableSharding("Airbnb");

sh.shardCollection("Airbnb.Listings", { property\_type: 1 } )

# Comandos ip forwarding

Sql: netsh interface portproxy add v4tov4 listenaddress = 25.5.150.109 listenport = 1233 connectaddress = 25.6.191.202 connectport = 1433

Monolítico:

netsh interface portproxy add v4tov4 listenaddress=25.6.191.202 listenport=50000 connectaddress=25.6.86.167 connectport=27016

# Comandos Logstash

docker run --rm -it -e XPACK\_MONITORING\_ENABLED=false -v C:\Users\kevin\OneDrive\Escritorio\Logstash\pipeline:/usr/share/logstash/pipeline/ -v C:\Users\kevin\OneDrive\Escritorio\Logstash\Listings.csv:/usr/share/logstash/file.csv -v C:\Users\kevin\OneDrive\Escritorio\Logstash\jdbc\sqljdbc\_9.4\enu\mssql-jdbc-9.4.0.jre8.jar:/usr/share/logstash/jdbc8.jar logstash:7.14.1 sh -c "bin/logstash-plugin install --version 3.1.5 logstash-output-mongodb && bin/logstash-plugin install logstash-output-jdbc && logstash -f /usr/share/logstash/pipeline/ "

# Config Logstash

input {

file {

path => "/usr/share/logstash/file.csv"

start\_position => "beginning"

}

}

filter {

csv {

separator => ","

columns => ["listing\_id","name","host\_id","host\_since","host\_location","host\_response\_time","host\_response\_rate","host\_acceptance\_rate","host\_is\_superhost","host\_total\_listings\_count","host\_has\_profile\_pic","host\_identity\_verified","neighbourhood","district","city","latitude","longitude","property\_type","room\_type","accommodates","bedrooms","amenities","price","minimum\_nights","maximum\_nights","review\_scores\_rating","review\_scores\_accuracy","review\_scores\_cleanliness","review\_scores\_checkin","review\_scores\_communication","review\_scores\_location","review\_scores\_value","instant\_bookable"]

remove\_field => ["message","host","path","host\_since","host\_location","host\_response\_time","host\_response\_rate","host\_acceptance\_rate","host\_is\_superhost","host\_total\_listings\_count","host\_has\_profile\_pic","host\_identity\_verified","latitude","longitude","accommodates","amenities","review\_scores\_accuracy","review\_scores\_cleanliness","review\_scores\_checkin","review\_scores\_communication","review\_scores\_location","review\_scores\_value","instant\_bookable"]

}

}

output {

jdbc {

id=> "sqllogstash"

driver\_jar\_path => "/usr/share/logstash/jdbc8.jar"

connection\_string => "jdbc:sqlserver://25.6.191.202:1433;

databasename=listings;

user=sa;

password=123456;"

statement => [ "INSERT INTO tb\_listings (listing\_id,name,host\_id,neighbourhood,district,city,property\_type,room\_type,bedrooms,price,minimum\_nights,maximum\_nights,review\_scores\_rating) VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?)", "listing\_id","name","host\_id","neighbourhood","district","city","property\_type","room\_type","bedrooms","price","minimum\_nights","maximum\_nights","review\_scores\_rating" ]

}

mongodb {

id => "monolitico"

uri => "mongodb://25.6.86.167:27016/Airbnb"

database => "Airbnb"

collection => "Listings"

codec => "json"

}

mongodb {

id => "cluster"

uri => "mongodb://25.6.191.202:27017/Airbnb"

database => "Airbnb"

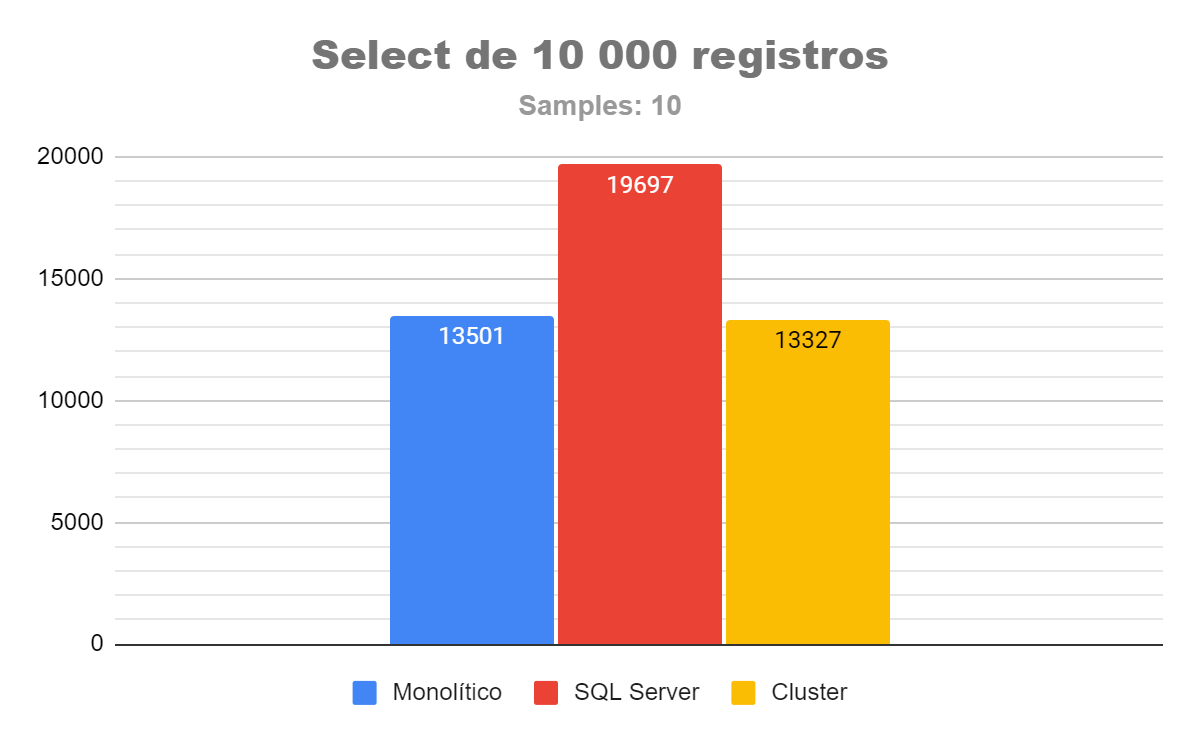
collection => "Listings"

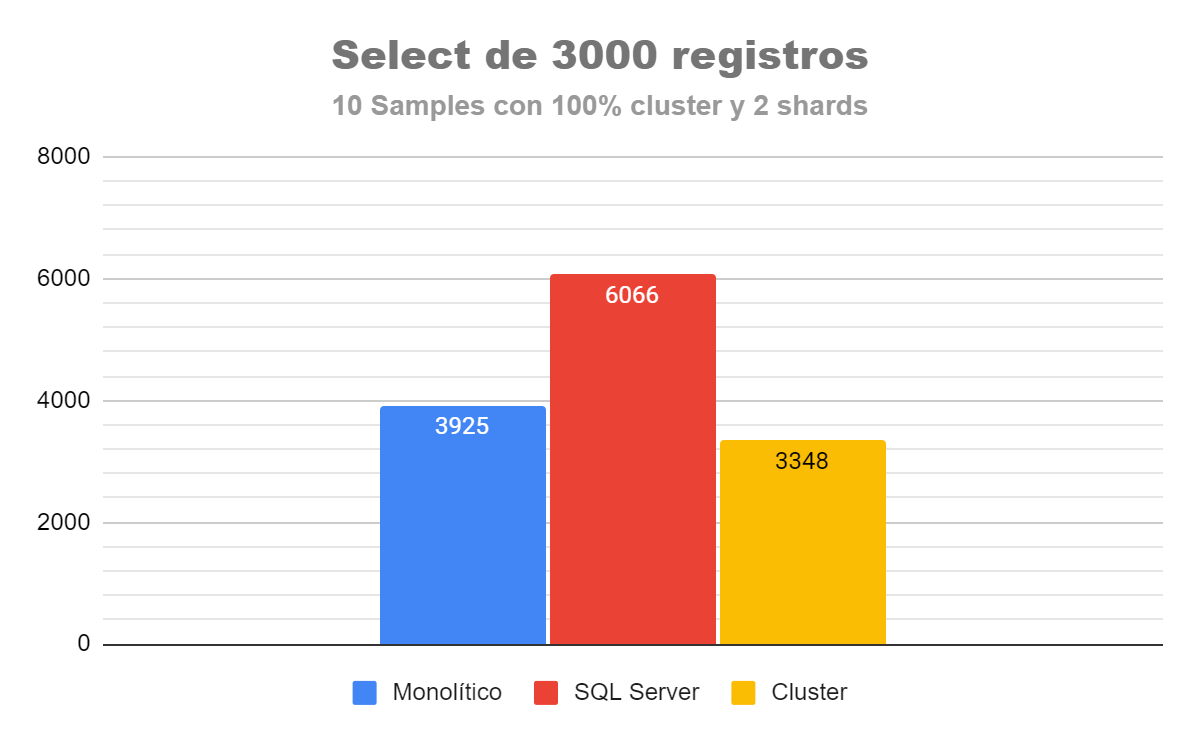
codec => "json"

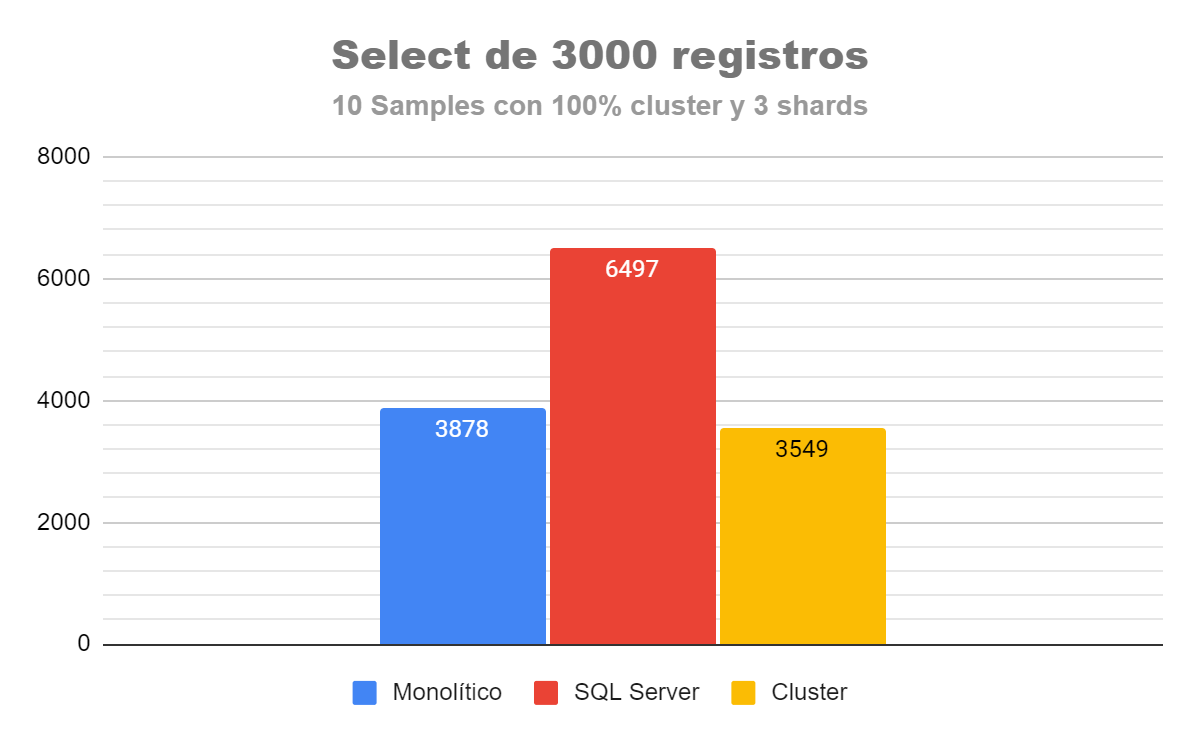
}

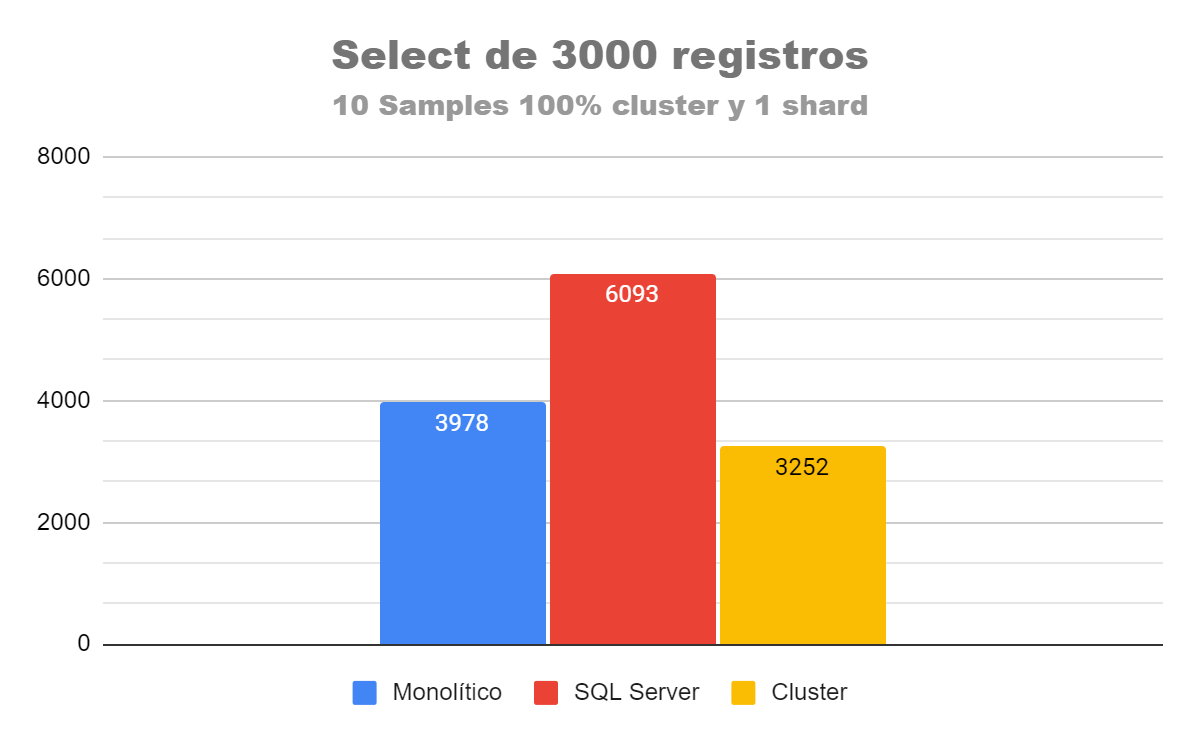
}

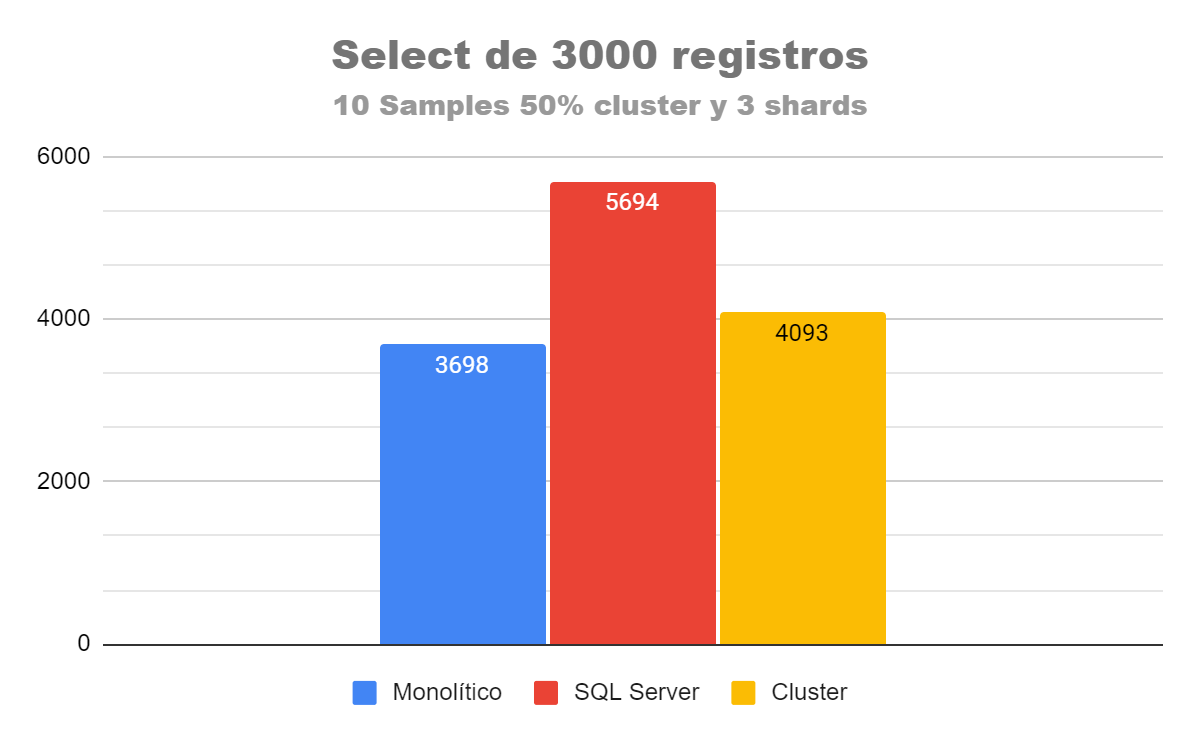
# Análisis de resultados

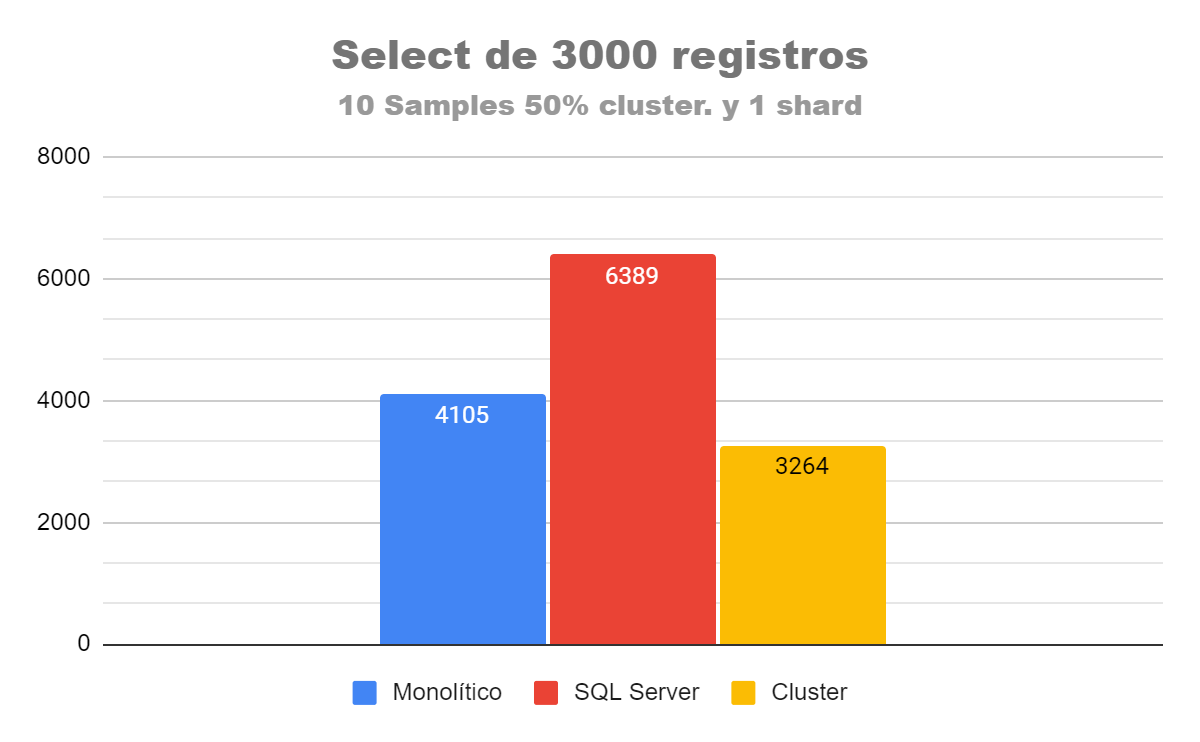


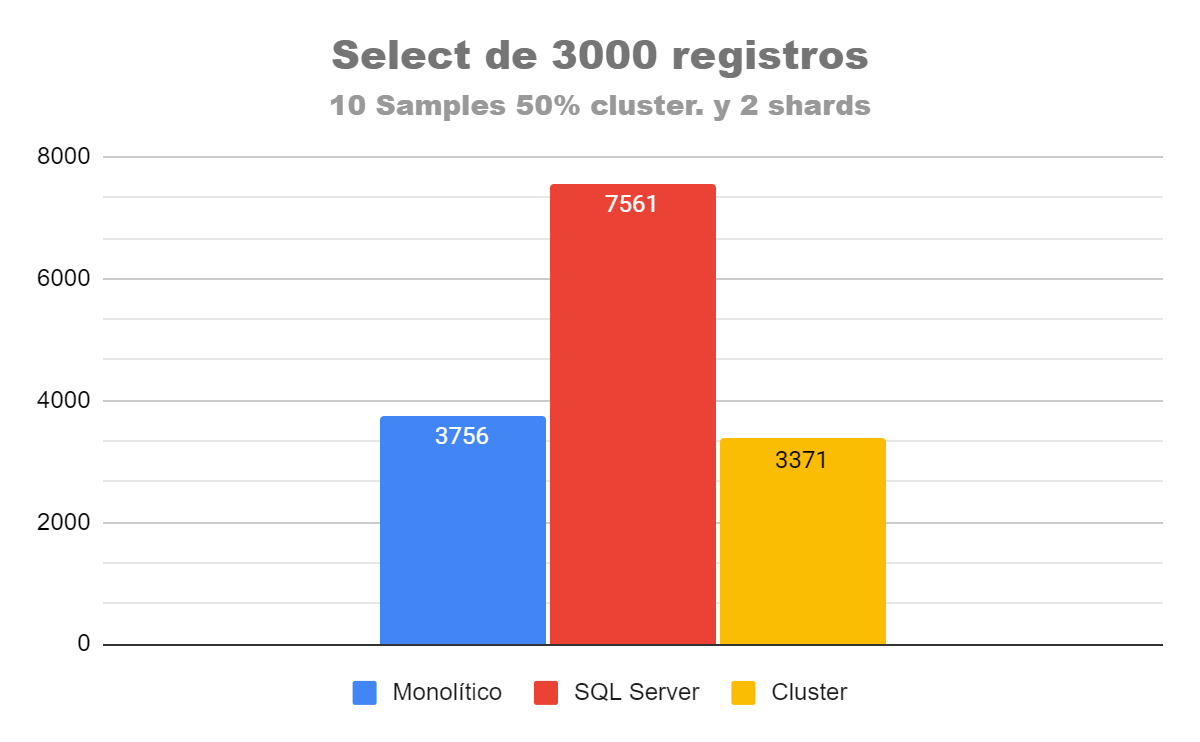












**Conclusiones**

Se realizó una primera prueba de un select de 10000 registros, y unas restantes seis pruebas de un select de 3000 registros con porcentajes diferentes del funcionamiento del cluster, donde un 50% de su funcionamiento (Se apagó el config 1, el shard 3.2, shard 2.1, shard 1.1) y se indican los shards consultados.

En la prueba #1 del select de 10000 registros, con el funcionamiento normal de todo el cluster, se observa un acercamiento de tiempo entre el Monolítico y el cluster, donde por una diferencia de 174 milisegundos, el Monolítico es más lento que el cluster.

En el 100% de las pruebas, SQL Server fue el que duró más. Solamente en la prueba #5, el cluster fue más lento a comparación del Monolítico, donde funcionaba a un 50% con 3 shards consultados. Por lo tanto, en un 87.5 % de las pruebas, el cluster fue el más rápido de los tres servidores.

La velocidad del cluster se ve afectada principalmente por el número de shards consultados en conjunto con un porcentaje del cluster apagado, donde, entre la prueba del 50% del cluster funcionando con un shard desactivado y el 50% del cluster funcionando con 2 shards desactivados, hay un incremento de 107 milisegundos, mientras que de la prueba con 50% del cluster funcionando con 2 shards consultados y la prueba con 50% del cluster con 3 shards consultados, hay un incremento de 722 de milisegundos.

Por su parte, en las pruebas donde solamente se desactivaron shards, entre la prueba con un shard desactivado y la prueba con dos shards hay una diferencia de 96 milisegundos, mientras que de esta última prueba mencionada y la prueba con 3 shards desactivados, hay un incremento de 201 milisegundos.