Ecosistema de Storage

Métricas Core

// Conceptos básicos

¿Qué son?

Las métricas core están medidas desde la perspectiva del usuario, nos explicita cuanto estamos afectando al usuario. Estas métricas miden la salud de los negocios principales de MELI y aplicaciones que generan afectación sobre estas métricas en caso de caída, tienen características de cuidado particulares.

Las apps con métricas core tienen, dependiendo del servicio, mejores o mayores garantías en términos de respaldos, tiempos de restore, alta disponibilidad, tiempos de respuesta, etc. También se debe tener presente la Experiencia de Usuario.

Cuando creo un servicio me pregunta si mi App afecta Métricas Core. Debo definirlas correctamente para evitar inconvenientes a futuro. Estas Apps llevan una codificación especial, y si defino erróneamente estas métricas, luego debo ocupar recursos migrandolo.

Cuando hay una afectación en la compañía, se priorizan aquellas apps que afectan métricas core.

Cuando hay un release se dejan para lo último las apps que afectan métricas core.

- Agregar los tags apropiados es **fundamental**

  Ejemplos de Métricas Core:

  - Cantidad de pagos
  - Cantidad de preguntas
  - Cantidad de Total listing (publicaciones)
  - Tiempo de demora en concretar un Envío

Mirrors / Slaves / Secondaries

- Algunos servicios tienen réplicas (habitualmente inconsistentes) con respecto a sus primarios.

- Esto agrega una capa de redundancia y alta disponibilidad, mientras que permite segmentar el tráfico según tipo o prioridad.

- Se puede redireccionar tráfico en caso de pérdida del master.

- Hay varios modelos en función de diferentes categorizaciones:
    - **Activo / Activo:** Cuando tengo una réplica los dos reciben tráfico productivo a la vez en distintos porcentajes. Las réplicas siempre están funcionando pero es más costoso. Ej: En ML el Key-Value MultiRegión es Activo-Activo, parte del tráfico va a una región, y la otra a la otra región.
    - **Activo / Pasivo:** Similar al anterior pero puede tener varias bifurcaciones. Si se cae la BD primaria, por cualquier eventualidad (catástrofes,etc), uso el secundario. Necesito probar con frecuencia con funciona, más allá de que no lleve tanto mantenimiento. Recordá que *“Lo que no se prueba, no funciona”*.
    - **Escritura-Lectura / Lectura:** Se puede segmentar de forma de que las escrituras vayan a un BD y la Lectura se haga en otra. Ej: Los User, escribe en Oracle y lee de un KVS. Las lecturas son más costosas que las escrituras. También puedo segmentar por importancia.
    - **Sync replica / Async replica:** Cuando se escribe en el Master y se réplica en el segundo. Tiene tiempos degradados de respuesta ya que debe asegurarse que se escribió todo bien para poder responder.
    - **Write local - read global / Write local - read local:** Replicas multi regiones. Ej: Pensando en videojuegos online, si tengo un servidor en Latinoamérica, quienes jueguen desde Asia van a tener una experiencia desagradable por la latencia; Entonces lo que se hace en estos casos es tener una réplica en el continente, que lea y escriba ahí para disminuir esta latencia. Luego, todo lo que está en Latinoamérica se replicará al servidor en Asia, y lo mismo con lo que se encuentra en Asia se replicará a Latinoamerica, en caso de Backup.
    - Etc

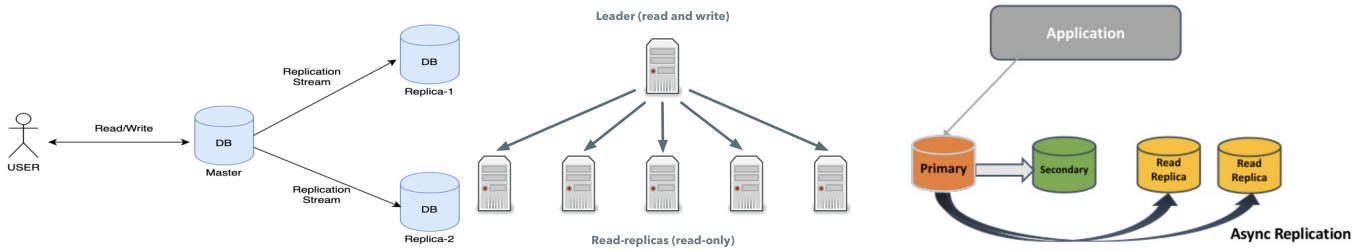

Imagen 1: Ejemplo de distintos modelos de réplicas utilizados en ML.

Backups / Restore

- Si bien podes hacer vos mismo backup en algunos servicios, en general se hacen automáticamente.
- KVS y bases de datos soportan PITR. Esto puede tener alguna pérdida de datos de 1 o 2 minutos (CDC podría ayudar).
- Object Storage no hace backup, pero tiene histórico de versión y cross region replication.
- DS hace backups una vez por día y guarda el diff en BigQueue para re-hidratar en caso de ser necesario. También la data está replicada en los mirrors.
- Audits también tiene cross region replication.

// Aspectos claves

- Es fundamental estar al tanto de los **tiempos de restore**. Si tu DB o servicio crece de forma no sana (demasiado o demasiado sobre una cierta partición) convendrá analizar si es mejor hacer un sharding porque las consultas y el restore serán lentos.