1. Migrar amb pvmove

Com moure mitjançant pvmove una LV existent a un altre PV

1. Comprovar primer que n'hi ha espai disponible al PV, executant comandes com: **pvs**, **vgs**, **lvs**
2. Executar **pvmove [disc1] [disc2]** si es volen passar tots els extents d'un disc a un altre. Si es vol moure únicament un LV existent de un PV concret llavors executem **pvmove –n [nom LV] [disc1] [disc2]**. Amb el paràmetre -n especifiquem quin LV es el que es vol desplaçar.
3. Comprovar que el traspàs s'ha realitzat correctament mitjançant la comanda **lvs -a -o +devices**

En fer aquest procediment en comptes de copiar fitxers manualment, s'evita haver de desmuntar el volum i aturar el servei, ja que **pvmove** realitza la migració de dades a nivell de blocs de manera segura. Això garanteix la integritat de les dades, manté els permisos i la configuració originals, i redueix el risc d'errors o pèrdua d'informació durant el trasllat.

2. Configurar mirall amb LVM

He configurat un mirall per al volum **/opt/webapp/data** amb l'ordre **lvconvert -m1 /dev/vg_webapp/lv_data**, creant dues còpies de les dades en discos diferents dins del grup de volums. Després de comprovar-ne l'estat amb **lvs -a -o +devices**, he simulat la fallada d'un disc amb **pvchange --offline /dev/nvme0n2** i les dades han continuat accessibles gràcies al mirall. En substituir el disc, el mirall es pot reparar amb **lvconvert --repair /dev/vg_webapp/lv_data**. Aquesta configuració garanteix una millora en lectura degut a que LVM por llegir de qualsevol copia però redueix el rendiment d'escriptura, ja que cada dada s'ha de duplicar, i requereix el doble d'espai d'emmagatzematge reduint així la capacitat màxima útil.

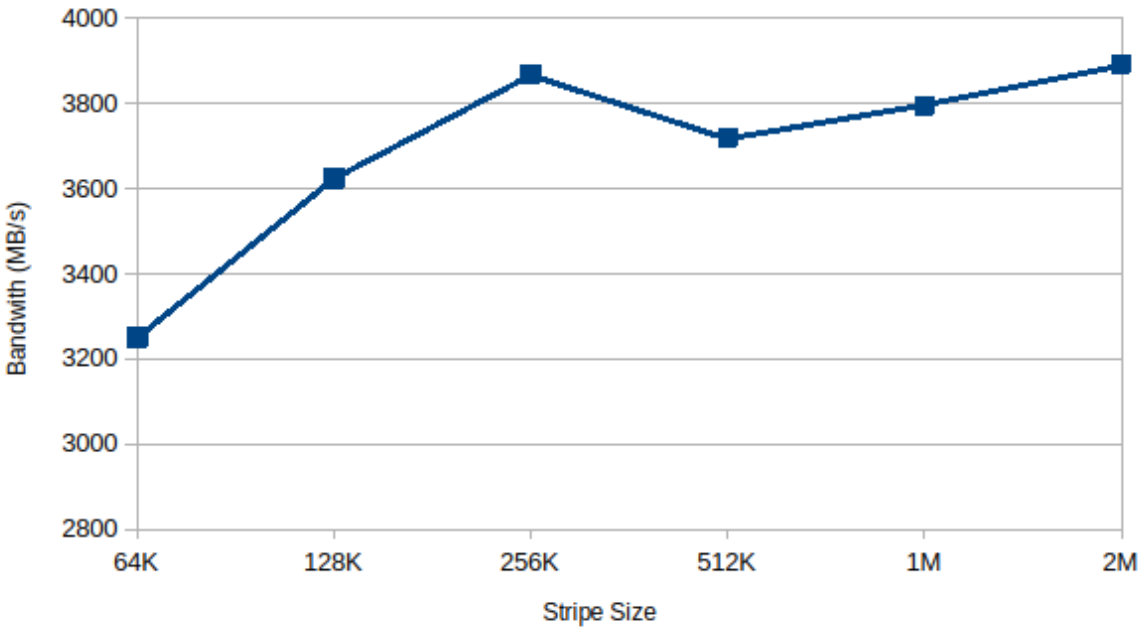
3. Avaluació de rendiment amb diverses mides d'stripe

Per a executar aquest experiment he creat aquests 2 scripts en bash per automatizar les proves:

```
#!/bin/bash

VG=vg_webapp

LV=lv_striping

amounts=(64K 128K 256K 512K 1M 2M)

for amount in "${amounts[@]}"; do

        sudo lvcreate -n $LV -i 2 -L 4G --stripe-size $amount vg_webapp

        sudo mkfs.ext4 /dev/$VG/$LV

        sudo mount /dev/$VG/$LV /mnt/striped

        fio --name=test_$amount \

        --directory=/mnt/striped \

        --size=1G \

        --rw=write \

        --bs=512k \

        --group_reporting \

        --output=resultat_$amount.txt

        umount /mnt/striped

        sudo lvremove /dev/$VG/$LV

done
```

```
#!/bin/bash

RESULT_DIR="./"

FILES=($RESULT_DIR/resultat_*.txt)

line=""

for file in "${FILES[@]}"; do

        size=$(basename "$file" .txt | cut -d'_' -f2)
        bw=$(grep "WRITE:" "$file" | sed -n 's/.*bw=.*(\([0-9]*\)MB\/s).*/\1/p')
        line="$line$size:$bw "

done

echo $line
```

He executat 3 vegades les diferents mides d'stripe per a 1GB de càrrega i he calculat la mitjana de rendiment entre les tres execucions ja que també la gràfica depèn de l'atzar. En generar la taula m'ha donat aquests resultats:



Veiem que en incrementar l'Stripe-size hi ha una millora bastant notable de l'ample de banda, especialment en el tram inicial. La gràfica mostra un salt de rendiment molt significatiu en passar de 64K (aproximadament 3250 MB/s) a 256K (aproximadament 3860 MB/s), que sembla ser un punt òptim inicial per a aquesta prova.

Tot i que la relació no és perfectament lineal i s'observa una lleugera caiguda a 512K, la tendència general és que mides de franja més grans afavoreixen l'ample de banda. El rendiment es recupera i continua pujant fins a assolir el valor màxim de tota la gràfica als 2M (prop de 3900 MB/s). Això suggereix que per a aquesta càrrega de treball específica, un cop superat el llindar dels 128K, els guanys de rendiment són menys pronunciats, però positius. Tota aquesta millora pot estar influenciada pot estar influenciada per el fet de que s'han de realitzar menys sincronitzacions entre els discs.