Universitat Autònoma de Barcelona

Dipòsit digital de documents de la UAB

This is the **published version** of the bachelor thesis:

del Viejo Avila, Ivan; Rexachs del Rosario, Dolores Isabel, dir. Anàlisi de prestacions de sistemes d'emmagatzematge. 2022. (958 Enginyeria Informàtica)

This version is available at https://ddd.uab.cat/record/264165

under the terms of the BY-NC-ND license

Anàlisi de prestacions de sistemes d'emmagatzematge en aplicacions de Intel·ligència Artificial

Iván Del Viejo

Resum– Els sistemes d'emmagatzematge secundari es caracteritzen per proporcionar un espai de memòria no volàtil i per tant persistent. Aquests sistemes juguen un paper essencial en el comportament de les aplicacions i el rendiment del sistema amb un us intensiu de peticions a disc com podien ser les aplicacions d'IA en les seves etapes d'entrenament i testing. En aquest treball s'ha fet un estudi de com podríem monitoritzar l'accés al sistema d'enmagatzematge de les aplicacions d'IA utilitzant una xarxa neural, quines eines a l'abast del usuari poden ser útils i quines mètriques poden proporcionar informació rellevant per a entendre el comportament del sistema, l'anàlisi de prestacions, com definim la freqüència de monitorització i com afecta aixo a la nostra precisió a l'hora de visualitzar gràficament les dades. Les proves s'han realitzat amb diferent mida de memòria física per tal de provocar *swap* i entendre aquest proces en el context del anàlisi de prestacions i com determinar el nombre de recursos necessaris que la nostra aplicació requereix.

Paraules clau– Monitorització, memòria física, swap, rendiment, xarxa neural .

Abstract– Secondary memory systems are characterised by providing a non-volatile and therefore persistent memory space. These systems play an essential role in the behaviour of the applications and the performance of the system with an intensive use of disk requests. In this work we have studied how we could monitor the disk access of AI applications using a neural network, which tools at the user's disposal can be useful and which metrics can provide relevant information for performance analysis, how we define the monitoring frequency and how this affects our accuracy in graphically visualising the data and understanding the behaviour of the system. The tests have been performed with different physical memory size in order to cause swap and understand this process in the context of performance analysis and how to determine the number of necessary resources that our application requires.
Keywords– Monitoring, physical memory, swap, performance, neural network.

1 Introducció - Context del treball

L'ENTRADA i sortida de dades es un factor determinant del rendiment en l'execució d'aplicacions. Al llarg dels últims anys s'han fet grans avenços en elements com la CPU o la RAM donant lloc a un augment considerable en la capacitat de comput d'aquests sistemes. En contraposició, no s'ha avançat al mateix ritme en l'àmbit dels dispositius d'emmagatzematge secundari. Aixo dona lloc a sistemes on el comput supera la barrera dels 15 Tflops/s, [10]però el ample de banda de E/S és d'uns MBytes/s.

Es primordial entendre el patró del accés a dades de disc, fer un us eficient dels recursos i identificar possibles problemes. En l'estudi del rendiment en l'accés a disc hem de tenir en compte sobre quina infraestructura d'emmagatzematge treballem(HHD, SSD, numero de discos, mida dels buffers...), també en funció del problema podem tenir un cas de *write-heavy* o *read-heavy*, el nivell de RAID que definim i molts altres factors que poden estar afectant al redniemnt.

En conseqüència, trobar una manera de optimitzar l'accés a dades en aquests entorns es critic per a reduir l'impacte d'aquest fenomen. En el procés de recerca de informació per tal de trobar quines alternatives existeixen per mi-

• E-mail de contacte: ivandelviejo@hotmail.es
• Mención realitzada: Enginyeria de Computadors
• Treball tutoritzat per: Dolores Rexachs (Departament d'Arquitectura de Computadors i Sistemes Operatius)
• Curs 2021/22

tigar el problema de E/S en la computació paral·lela podem veure que el concepte *parallel I/O* es repeteix. La idea que hi ha darrera aquesta tècnica es realitzar múltiples operacions d'entrada i sortida de manera simultània, en comptes de fer-ho en serie.

Les aplicacions de intel·ligència artificial tenen grans requisits pel que fa a l'entrada i sortida de dades, sobretot en la etapa de entrenament que es on cal accedir a una major quantitat de dades, i poden suposar factor limitant de rendiment. En aquest tipus d'aplicacions tenim un fase de *training* i una de *testing*. Un gran percentatge del dataset original esta destinat a la etapa d'entrenament, la resta s'utilitzara en l'etapa de validació. *Machine Learning* es la branca de la intel·ligència artificial que engloba aquesta mena d'algorismes.

Aquest procés s'anomena *cross-validation* i funciona tal i com es mostra a la figura 1. El conjunt de *benchmarks* que s'utilitzen en aquest estudi implementen aquest proces. A partir d'un *dataset* inicial, definim els subconjunts de *test* i *train* esmentats anteriorment. L'entrenament es realitza tenint en compte la variable *label* o variable objectiu, es a dir, la variable que el nostre model ha de poder predir. Naturalment, la següent etapa es realitza únicament amb les dades del testing-set. Una vegada hem obtingut els resultats que el model ofereix, els comparem amb els esperats(label). Es així com es calcula l'*accuracy* del nostre model.

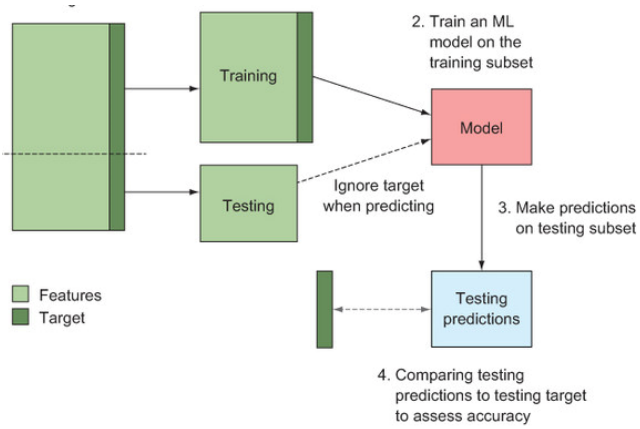

Fig. 1: Cross-validation

2 Marc teòric

En la següent secció s'introdueix al lector a alguns conceptes tècnics que s'utilitzen al llarg del treball.

2.1 Patró d'accés

El patró d'accés fa referencia a la manera en que un sistema o programa llegeix o escriu en el dispositiu de emmagatzematge[7]. Els diferents patrons que existeixen difereixen en el nivell de localitat. Entenem per nivell de localitat o principi de localitat com la tendència del processador a accedir a posicions de memòria consecutives, aquest concepte es coneix com a localitat espacial. La localitat temporal no es te en compte quan treballem amb l'emmagatzemament secundari ja que aquest esta pensat per ser accedit una sola vegada i no repetidament en el mateix espai de temps.

El patró més sencill es el seqüencial, en aquest cas les dades son llegides, processades i escrites de manera incremental o decremental en posicions contigües en memòria. L'accés strided es semblant al seqüencial amb la diferencia de que els accessos a memòria estan distanciats n posicions, si el *stride* es massa gran pot dificultar l'efectivitat dels mecanismes de *caching* i *prefetching*. Per ultim l'accés aleatori dificulta l'aplicació de tècniques de millora de rendiment i comporta ineficiències.

2.2 Swap

Swap es produeix una vegada ens quedem sense memòria física, i per tant, utilitzem memòria virtual, aquesta emmagatzema les dades en disc. Com be sabem, llegir de disc es molt més lent que llegir de memòria principal, per tant, aquest fet redueix el rendiment. El procés de intercanviar dades entre memòria virtual i física ho anomenem *swap*, l'espai en disc es coneix per ant com a *swap space*. La mida del *swap space* ve determinada per la mida del nostre dispositiu de emmagatzematge, a major capacitat més espai per aquest intercanvi. Aquest espai es modificable segons les nostres necessitats.

Els diferents processos que corren en el nostre sistema, poden ser temporalment traslladats de memòria principal a memòria secundaria, així deixem aquests espai lliure per a altres processos. L'objectiu d'aquest proces és augmentar la utilització de la memòria. El concepte de *swap* el podem desglossar en dos de diferents: *swap-in* i *swap-out*. El primer es el mètode encarregat de moure un process del disc i emmagatzemar-lo en memoria principal, *swap* out fa referencia al proces invers.

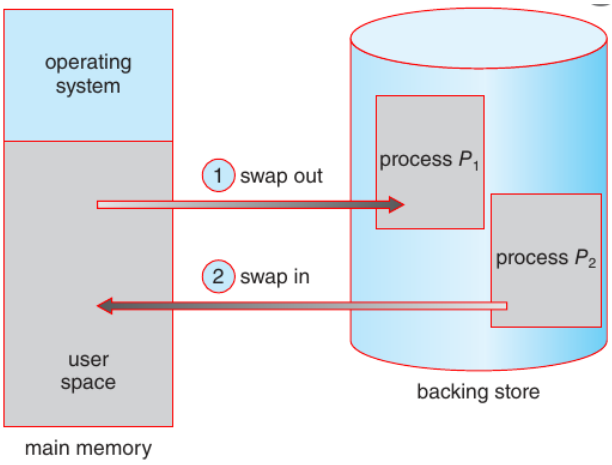

Fig. 2: Swap: swap-out i swap-in

3 Objectiu General

Aquest estudi te com a finalitat fer un anàlisi del comportament de les entrades/sortides (E/S) de aplicacions amb gran quantitat d'accés a dades i proporcionar informació sobre la presa de decisions. Per aixo cal identificar els patrons d'accés d'E/S de les aplicacions i analitzar les prestacions

fent una cerca de la relació entre el patró d'accés de les aplicacions al sistema d'emmagatzematge i les diferents configuracions de sistema pel que fa a la memòria. Un dels objectius principals es poder representar de manera gràfica les grans quantitats de dades generades per les eines de monitorització i d'aquesta manera proporcionar al usuari major facilitat en l'anàlisi i en la presa de decisions sobre les necessitats del seu sistema.

3.1 Objectius específics inicials

- Definir l'entorn de treball sobre el que realitzem les proves i les seves possibilitats de configuració pel que fa al sistema de fitxers(RAID, sistemes de fitxers compartits, quantitat de memòria...).
- Cercar i analitzar una serie de eines de monitorització per tal de definir el subconjunt d'aquestes que ens permetran dur a terme l'anàlisi de rendiment corresponent.
- Obtenir una visió amplia de les tècniques o eines d'anàlisi de rendiment i identificar les mètriques que son importants per l'anàlisi de l'E/S.
- Implementar i executar *benchmarks* fent ús de múltiples processos que accedeixen al sistema d'emmagatzematge per tal de monitoritzar les prestacions del sistema.
- Utilitzar diverses eines d'observabilitat per tal de dur a terme les tasques de monitorització i visualització de mètriques de rendiment.
- Analitzar els resultats obtinguts a les proves realitzades, en les que es modificaran diferents paràmetres, així com la mida de la memòria, el patró d'accés a dades o el numero de processos s'executen al sistema. En aquest punt cal extraure conclusions per tal de generar un report final del estudi de prestacions.

3.2 Entorn de treball

Els tests de rendiment que composen el nostre estudi es realitzaran en maquines virtuals. Farem us del software Oracle VM VirtualBox amb una distribució Linux, concretament Debian, la versió de Debian sobre la que treballarem es la 11.2.0. El motiu de fer us d'una distribució Linux, es que aquestes compten amb una serie d'eines que permeten fer un anàlisi de rendiment del sistema. A més a més, les maquines virtuals permeten configurar la quantitat de memòria física assignada, per tal de fer proves amb diferents valors.

4 EINES DE MONITORITZACIÓ

Com hem esmentat anteriorment, per a dur a terme el nostre estudi cal utilitzar una serie de eines que ens permeten obtenir dades de l'execució del *benchmark*. Les diferents mètriques formaran el conjunt de dades que ens seran útils per a fer un anàlisi de les operacions d'E/S i poder extraure conclusions envers els possibles *bottlenceks* de l'execució i la identificació de ineficiències.

El nostre entorn de proves, com la majoria de distribucions linux, ja compta amb una serie de eines de monitorització de prestacions que permeten obtenir dades a diferents nivells, com be podria ser la xarxa, la CPU o tant el sistema de fitxers com l'accés a disc(iotop,iostat,top) i operacions de *swap*(swapon) que son les que en el nostre cas més ens poden interessar. Per a aprofundir en els processos de accés al sistema d'E/S s'utilitzara Darshan.
A la figura 3 podem veure ressaltades aquelles eines de ser susceptibles de ser utilitzades al llarg del estudi.

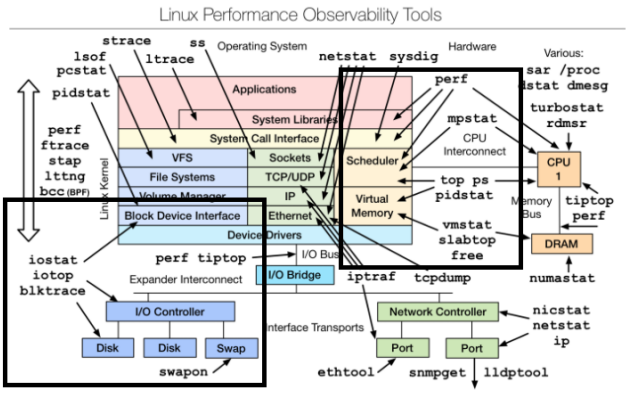

Fig. 3: Linux Performance Observability Tools

A continuació descriurem la utilitat d'aquest subconjunt d'eines i com pensem que el seu us pot ser profitós pel desenvolupament de la investigació.

4.1 Iotop

A través de iotop tal i com es mosta a la figura 5 obtenim l'ample de banda d'E/S llegit i escrit durant el proces de mostreig per cadascun dels processos esmentats. També mostra el percentatge de temps que aquest proces a estat en espera o realitzant *swap*. Les dades es mostren a temps real, pel que permet al usuari entendre amb major profunditat com evoluciona l'ús del accés a disc.

Com hem dit, iotop mostra tots els processos però si volem monitoritzar nomes aquells processos que realitzen operacions de I/O podem utilitzar el *flag* -o.

```
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN     IO>    COMMAND
1809 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.02 % [kworker~4-events]
   1 be/4 root        0.00 B/s    0.00 B/s  0.00 %  0.00 % init
```

Fig. 4: Output de l'eina iotop

Les mètriques que són potencialment útils estan descrites en el apèndix del document.

4.2 Iostat

La comanda iostat s'utilitza per a supervisar la carga del sistema d'entrada/sortida. Iostat es part del paquet sysstat. Es interessant utilitzar aquesta eina ja que ens proporciona informes que proporcionen la informació necessària com per a realitzar canvis en la configuració del sistema per tal d'equilibrar la carrega entre els discos físics.

Aquesta eina proporciona tants reports de rendiment de CPU com de els diferents dispositius d'emmagatzematge. Per a que ens mostri la informació referent als dispositius d'emmagatzematge hem de utilitzar la opció -d".

```
Linux 5.10.0-12-amd64 (debian)  06/04/22        _x86_64_        (1 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
          16,09    0,06    3,09   15,50    0,00   65,26

Device             tps    kB_read/s    kB_wrtn/s    kB_dscd/s    kB_read    kB_wrtn    kB_dscd
sda             139,86      2950,92      2452,83         0,00    5823012    4840148          0
sr0               0,01         0,00         0,00         0,00          2          0          0
sr1               0,02         0,53         0,00         0,00       1042          0          0
```

Fig. 5: Output de l'eina iostat

Les mètriques que són potencialment útils per al nostre estudi estan descrites en el apèndix del document.

4.3 Top

L'eina top proporciona una vista dinàmica a temps real del sistema. Proporciona una llista de processos i *threads*, mostra l'us de recursos a nivell de cpu, informació del us de memòria física i virtual i altres dades d'interès. A la figura 6 podem observar a la primera meitat de la imatge les mètriques pertinents al global de us de recursos en el sistema, mentre que a la llista de la part inferior ho tenim desglossat segons el proces en qüestió.

```
%Cpu(s):  3,3 us,  2,5 sy,  0,0 ni, 94,0 id,  0,0 wa,  0,0 hi,  0,2 si,  0,0 st
MiB Mem :   2521,7 total,   1167,0 free,    664,9 used,    689,8 buff/cache
MiB Swap:    975,0 total,    975,0 free,      0,0 used.   1691,7 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   2260 _apt      20   0   21804   9600   8604 S  13,6   0,4   0:02.37 http
    431 message+  20   0    9776   6004   4016 S   0,3   0,2   0:01.94 dbus-da+
   1074 ivan      20   0 3892732 269512 119592 S   0,3  10,4   0:07.57 gnome-s+
   1192 ivan      20   0  790960  97312  48232 S   0,3   3,8   0:05.92 gnome-s+
```

Fig. 6: Output eina Top

5 Característiques del sistema i anàlisi preliminar

Abans de realitzar les proves de rendiment amb una aplicació real cal tenir en compte les especificacions del sistema d'emmagatzematge amb el que treballem. L'emmagatzematge secundari de la nostra maquina compta amb un disc HDD de 8,6 GB de capacitat. Com hem comentat en apartats anteriors i com veiem a la figura 7 les zones de memòria estan dividides en particions, una d'aquestes es la partició destinada a *swap*. Aquesta serà utilitzada quan la memòria física es vegi saturada.

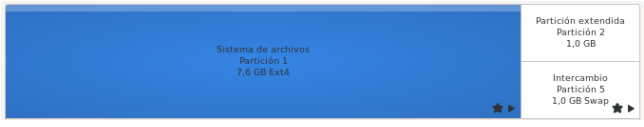

Fig. 7: Disc: particions i mida

Linux permet realitzar un *benchmark* senzill per conèixer algunes dades de rendiment del teu sistema d'emmagatzematge. Pensem que aquestes dades podrien ser útils més endavant a l'hora d'analitzar el rendiment amb la aplicació real. En aquesta prova de rendiment de disc es mesura la tassa mitjana d'accés i l'ample de banda de lectura. Hem realitzat dues proves, la primera d'elles amb 100 mostres per a l'ample de banda de lectura i 1000 per calcular el temps mitja d'accés.L'ample de banda de lectura ha sigut de 149.2 MB/s de mitja mentre que el temps d'accés mitja ha estat de 7,87 msec.

La segona proba s'ha realitzat augmentant el numero de mostres per tal d'obtenir un resultat mes acurat, d'aquesta manera ens assegurem que les dades obtingudes en el *benchamark* anterior no ha estat afectat pel possible soroll. Els resultats obtinguts han estat lleugerament millors, l'ample de banda mitja ha sigut de 174,4 MB/s i el temps d'accés mitja de 7,71 mseg.

La gràfica blau correspon a l'ample de banda, mesurat amb la columna de l'esquerra en MB/s. Els punts verds fan referencia al temps d'accés i es mesura amb l'eix y de la dreta en milisegons.

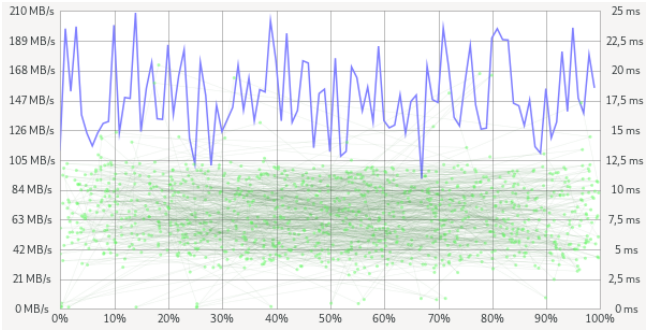

Fig. 8: Proba de rendiment de disc

6 Anàlisi d'una aplicació

En aquest punt ja tenim un entorn raonablement complert on treballar i poder fer proves.

La aplicació que hem utilitzat es una xarxa neural escrita en python. Aquesta segueix el mètode conegut com *corss-validation* que es descriu en el segon apartat d'aquest document. Les xarxes neurals com aquestes pretenen simular el funcionament de una neurona humana. Aquesta simulació de neurona l'anomenem perceptró. Un perceptró te una serie de entrades, un nucli i una sortida. En funció de les entrades, si la suma d'aquestes supera un *threshold* aquesta s'activa i produeix un valor de sortida. A cadascuna de les entrades podem assignar pesos per determinar la importància de la entrada. Aquests algorismes pretenen trobar el millor valor dels pesos per a determinar si una neurona s'activa o no. D'aquesta manera formem xarxes amb múltiples capes que al llarg del entrenament actualitzen els seus pesos per a reduir l'error del model. En la següent imatge veiem una representació d'una neurona i de la seva funció d'activació que depèn de les entrades.

La aplicació utilitzada té una classe anomenada NuralNetwork que es l'encarregada de la gestió de aquesta xarxa: inicialitza les matrius de pesos, entrenament de la xarxa i actualització dels pesos amb la funció *train()*, execució del test amb la funció *run()*.

Al inici del codi es carreguen les imatges del *test-set* i el *training-set* que tenim al nostre sistema en un arxiu .csv. La base de dades utilitzada es la coneguda MNIST,

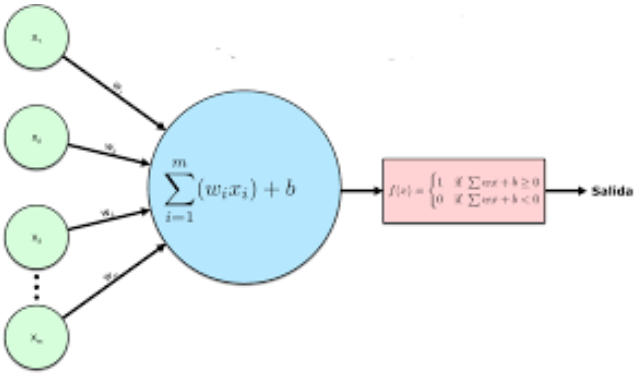

Fig. 9: Perceptró

aquesta esta composta per un conjunt de imatges cadascuna representada en una línia. Les imatges son un conjunt de 785 números entre 0 i 255. El primer numero de cada línia es el *label* o variable objectiu que caldrà predir, es a dir, el digit que representa la imatge. La resta de números representen la imatge de mida 28x28.

Amb la instrucció train_data.shape i test_data.shape podem veure el numero de imatges que conte cadascun dels datasets. Com podem veure el primer d'ells conta amb unes 60.000 imatges, m'entre que el de test tan sols unes 10.000.

```
Mida del training-set:
(60000, 785)
Mida del testing-set:
(10000, 785)
```

Fig. 10: Mida del training-set i test-set

El temps d'execució serà mesurat amb la eina *time*. Aquest temps es el temps d'execució base del programa, en fer tasques de monitorització on altres processos utilitzen recursos fa que el temps pugui augmentar. En qualsevol projecte d'aquestes característiques, cal tenir en compte el soroll que poden provocar aquestes eines a l'hora de mesurar les diverses mètriques de rendiment. En el nostre cas tant en la eina iotop com amb iostat on fem escriptures en un fitxer amb l'output no obtenim un impacte significatiu en la execució del programa.

6.1 Freqüència de mostreig: problemàtica i solució proposada

Aquest apartat pretén descriure la problemàtica amb el tractament de grans quantitats de dades generades en processos de monitorització de prestacions. A més a més mostar una possible solució que pot ser de gran utilitat per al usuari.

La freqüència de mostreig fa referencia a l'interval de temps entre una mesura i l'altre. Les eines utilitzades en aquesta primera etapa d'anàlisi com *iostat* proporcionen informació cada cert temps sobre diferents paràmetres del sistema. Aquesta informació pot ser mostrada en consola, però es recomanable redirigir-la cap a un altre fitxer per al seu posterior tractament.

En programes o aplicacions mes grans, com a les aplicacions d'IA que tractem en aquest treball on el numero d'accesos i el temps d'execució creix ràpidament, la quantitat de dades generades durant aquests temps es molt gran, el que fa que les tasques d'anàlisi de resultats puguin ser costoses en temps i esforç. Si imaginem aquesta situació amb programes molt més grans, a mesura que el temps d'execució augmenta, també ho fa la mida del fitxer de dades generat en la recol·lecció de dades de monitorització.

En aquesta situació podríem reduir la freqüència de mostreig per a reduir la quantitat de dades a analitzar, però aixo podria impactar la precisió de la mesura, perdent dades potencialment útils. La opció que sembla raonable es poder mostrar de manera gràfica les dades recollides per la eina *iostat* i poder així extraure conclusions de manera molt més rapida. A més a més pot facilitar molt la feina de comparar diferents mètriques ja que en comparar ambdues gràfiques podem veure si sembla haver-hi cap relació entre elles.

Hem utilitzat uns scripts que fan servir gnuplot, aquesta eina permet generar gràfiques de dues i tres dimensions a partir d'unes dades. A les referencies trobem el repositori github[6] d'on hem obtingut els scripts que ens ajuden a generar les imatges. Aquests scripts estaven pensats per a versions més antigues de iostat, hem readaptat aquests scripts segons les nostres necessitats.

El primer que fem es redireccionar la sortida de la comanda iostat cap a un fitxer auxiliar. A més a més de la informació pròpia de iostat afegim un *timestamp* per a poder representar la sortida en funció del temps quan generem les gràfiques. La comanda utilitzada es mostra a la figura següent. En aquest cas hem definit que la mesura es prengui cada 3 segons amb la opció -t amb la que iostat ens permet imprimir el *timestamp* de la mesura. La opció -xk permet mostrar un major numero de mètriques de rendiment en kilobytes.

```
iostat -xk -t 5 | awk '// {print strftime("%Y-%m-%d %H:%M:%S"),$0}' > iostat-server1.out
```

Fig. 11: Comanda iostat

En aquest punt obtenim un fitxer de dades que cal tractar i que serà utilitzat per generar la gràfica. El format del output obtingut es mostra a la següent figura.

```
2022-04-19 20:32:43 Device            r/s     rkB/s   rrqm/s  %rrqm r_await rareq-sz     w/s
2022-04-19 20:32:43 sda           826,80  12956,00  2412,20  74,47    1,17    15,67  409,00
2022-04-19 20:32:43 sr0             0,00      0,00     0,00   0,00    0,00     0,00    0,00
2022-04-19 20:32:43 sr1             0,00      0,00     0,00   0,00    0,00     0,00    0,00
2022-04-19 20:32:43
```

Fig. 12: Output obtingut amb iostat

Per a veure l'impacte i la importància de la elecció d'una freqüència de monitorització que proporcioni dades suficientment acurades per a realitzar una anàlisi correcte mostrarem un exemple de mètrica mostrejada cada 3 segons i a la mateixa execució mostrejada cada 6 segons.

Si observem la figura 13 podem veure la manera en la que aquest canvi en la freqüència de monitorització provoca una representació del comportament diferent. A la imatge de la dreta on la freqüència es de 3 segons veiem que el mostreig no proporciona la suficient precisió com per a diferenciar

les dues lectures de fitxers, com veiem a la figura de l'esquerra al llarg del minut 14:58 hi ha hagut un moment on no hi havien lectures a disc, però aixo la gràfica de la dreta no ho mostra.

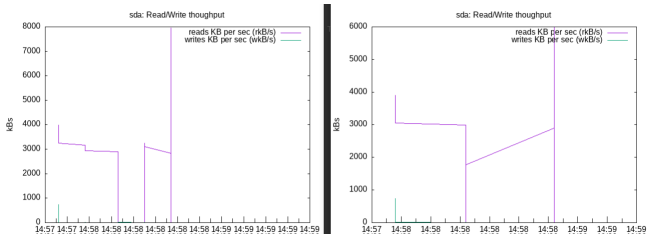

Fig. 13: Mostreig obtingut segons la freqüència

Un altre exemple de quines diferencies podem obtenir en els resultats fa referencia als valors màxims d'utilització d'una mètrica mesurada, com es pot veure a la figura 14 veiem que el mostreig de la dreta fa la mesura quan la mètrica mesurada arriba a 300, per altre banda l'exemple de l'esquerre fa altres mesures i detecta el pic fins els 700, com els sistemes informàtics canvien ràpidament ja que treballen a un ritme molt elevat si ho comparem amb l'escala amb que nosaltres monitoritzem un segon pot significar un augment considerable.

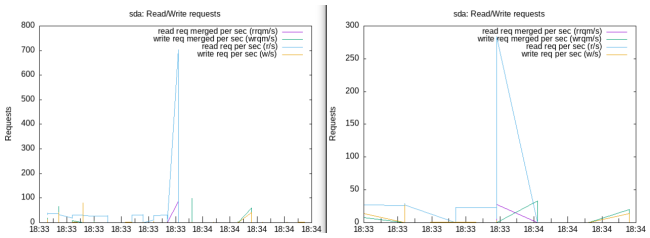

Fig. 14: Mostreig obtingut segons la freqüència

6.2 Resultats amb 4096 MB de memòria física

En aquest apartat volem representar un entorn no afectat per altres factors com pugui ser el swap i poder entendre l'impacte de la aplicació en disc únicament, sense que intervinguin actors externs, per aixo assignem una quantitat de memòria física elevada per a les necessitats de memòria de la nostre aplicació de intel·ligència artificial, per tant necessitem que les estructures de dades pròpies de la xarxa neural generada càpiga en RAM. Després de realitzar una serie de execucions i utilitzant l'eina *time* de *python* hem obtingut un temps d'execució de 111 segons de mitjana.

Aquests exemples no son gaire profitósos en termes de conclusions de rendiment del sistema d'emmagatzematge degut a que únicament s'accedeix en la lectura dels fitxers inicials, aquest es un entorn que podríem trobar en un entorn de producció real on la xarxa neural esta limitada per comput, aquest concepte es tractara en profunditat en la discussió dels resultats, però com hem comentat, ens serveix com a estàndard del comportament de la aplicació per defecte. Per aquest motiu en alguns aspectes s'entrarà en major detall en apartats posteriors on la quantitat de recursos a nivell de memòria física es menor, permetent-nos extraure conclusions del comportament del sistema de memòria secundari quan es troba sota pressió.

En aquest entorn hem de tenir especial cura amb algunes mètriques que poden ser afectades en gran mesura per el soroll degut a la relativament poca activitat en disc degut a que les dades caben en RAM. Es per aixo que algunes mètriques son susceptibles de no ser realment representatives o tant útils com en apartats posteriors, on la memòria es mes reduïda i es veu el impacte d'aquest fenomen reflexat a les mètriques. Per exemple, en un entorn on el temps de resposta sigui anormalment elevat degut a una saturació del recurs es fàcil veure aquest impacte, en un entorn on cap problema altera el temps de resposta, aquest es troba en valors molt baixos i per tant qualsevol petita alteració deguda al soroll provocat per altres processos podria afectar molt a la gràfica obtinguda.

Al final de les proves amb la eina iostat hem utilitzat iotop per conèixer el percentatge de *swap* i la quantitat de memòria llegida de disc. En aquest apartat el gran número de recursos permet obtenir un percentatge de *swap* de 0 al llarg de la execució que seria el desitjat en un entorn real. S'han llegit 153 MB de disc. Aquestes dades cal tenir-les en compte, en els exemples amb 2048 MB i 1024 MB de memòria física dedicarem un subapartat al *swap* i a la discussió final dels resultats compararem aquestes dades i com afecta a les lectures i escriptures a disc.

6.2.1 Utilització del dispositiu

A la figura 15 podem observar la utilització del disc al llarg de la execució. Com podem veure podem identificar els dos grans períodes D'accés a dades, el primer màxim corresponent al accés a les imatges de entrenament i el segon a les de test. Ambdós períodes son molt semblants diferenciant un pic en un petit instant en el període de test que pot haver sigut causat per intrusisme de altres processos accedint al disc, encara i aixi podem veure que el patró en ambdós pics son gairebé idèntics sobrepassant lleugerament el 20% d'utilització del dispositiu.

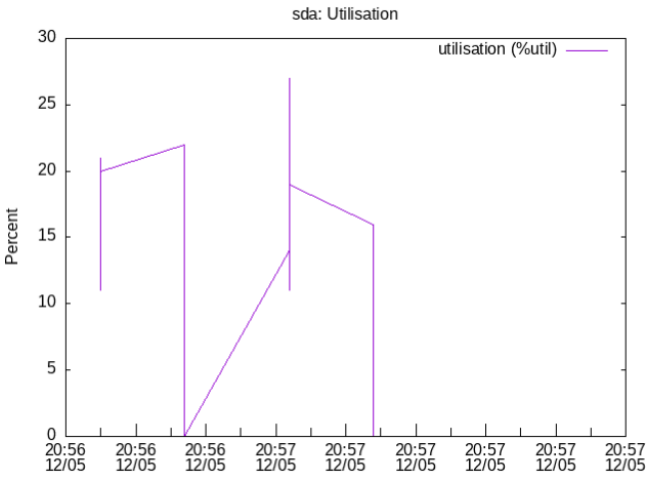

Fig. 15: %util del disc sda amb 4096 MB de memòria física

6.2.2 Peticions d'operacions d'E/S al dispositiu

A a següent gràfica observem les mètriques referents a les peticions al disc completades per segon, tant escriptura(w/s) com lectura (r/s). Juntament amb rrqm/s i wrqm/s per a mostrar les peticions que s'han fet *merge* abans de enviar-les al *device*. Aquestes mètriques seran útils més endavant, a l'hora de tractar el *swap* i com afecta a les operacions tant de entrada com de sortida al disc, concretament a la zona d'intercanvi. A més a més, ens serviran per mesurar l'augment de les lectures i escriptures quan la xarxa neural no cap en memòria principal. En aquest cas podem veure que el numero de peticions de lectura per segon que el disc efectua son molt més elevades que els escriptures ja que es el que la aplicació pròpiament fa. El que s'observa es que el disc realitza més ràpid les operacions del test que son molt menys que les del entrenament encara i que aquest màxim de velocitat de servei de petició es dona nomes en un moment molt concret de la execució i no es pas un efecte sostingut en el temps.

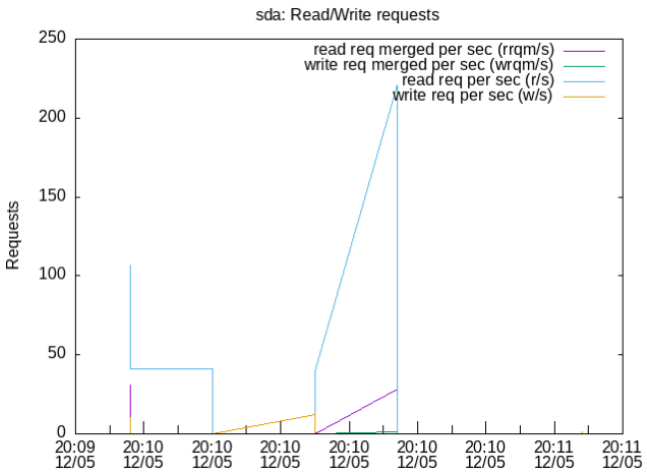

Fig. 16: Peticions d'E/S al dispositiu

6.2.3 Temps de servei de peticions i espera en cua

Amb valors elevats de recursos de memòria, aquesta mètrica ens serveix d'exemple de com en alguns casos algunes mètriques poden no ser representatives.

El que succeeix en aquest cas es que com el disc pot suplir amb molta facilitat les necessitats de peticions de la aplicació, la velocitat de resposta, mes concretament el temps en cua que es nul com veurem més endavant i el temps de servei, son molt baixos. En un entorn on l'aplicació mitjançant les seves crides intensives al dispositiu d'E/S provoqui retard en el temps de servei de les peticions, es fàcil observar com afecta ja que el soroll representara una porció mínima en el global.

Degut a que en aquest exemple el temps es tant baix, es susceptible de ser alterat per soroll provocat per altres processos, cal recordar que el disc es un element compartit i que les eines no aïllen únicament el comportament provocat per el nostre programa, sinó que recullen totes les dades. D'aquesta manera s'ha observat a les gràfiques una variància molt elevada i per tant presenten dades poc representatives.

6.2.4 Mida de la cua

Aquesta mètrica ve directament relacionada amb l'anterior, en aquest cas el disc rep peticions i al mateix ritme que les rep les serveix, en cap cas es necessari posar-las en cua, almenys no el temps suficient com per a que la eina pugui detectar la presencia de peticions a la cua.

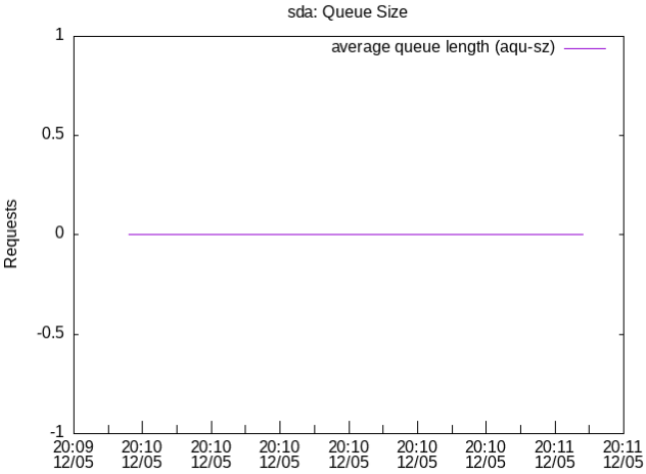

Fig. 17: Mida de la cua amb 4096 MB

6.2.5 Throughput

Per ultim mostrem el ritme al que el disc treballa segons les peticions que rep de la aplicació. Com podem veure a la figura 18 el ritme de treball no es elevat. En els pics d'us treballa per sota a 10 MB/s que es molt inferior a les possibilitats del disc, es a dir del valor mostrat de referencia al apartat 6 d'aquest treball(174 MB/s).

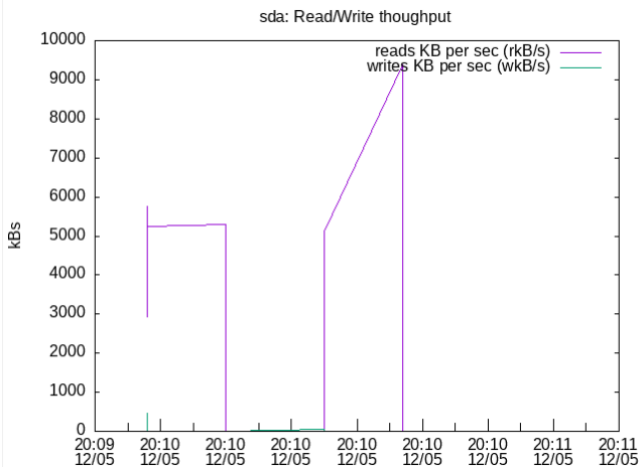

Fig. 18: Throughput amb 4096 MB

6.3 Resultats amb 2048 MB de memòria física

Aquest apartat te com a objectiu mostrar el rendiment ja afectat per un percentatge de *swap* considerable, com es veuen afectades les mètriques de rendiment, la correlació entre aquestes i quines conclusions podem extraure.

6.3.1 Utilització del dispositiu

En primer lloc visualitzarem la utilització del nostre *device*. Com es pot veure a la figura 19, el percentatge d'utilització del disc sda es molt elevat en alguns punts crítics al llarg de la execució, arribant al anomenat punt de saturació, però en la resta de moments el sistema no es troba sota pressió. Es habitual assumir que a quan més ens apropem al 100% d'utilització més saturat esta. Aixo es cert en aquest cas ja que utilitzem un HDD que serveix peticions d'una en una. Si l'usuari que fa les proves esta en un sistema amb una matriu de discos o RAID podria servir diverses peticions al mateix temps. En aquests dispositius aquesta mètrica simplement mostra el percentatge de temps que el dispositiu esta servint almenys una petició.

En el cas en que en molts punts d'execució de l'aplicació al punt de saturació del dispositiu o molt proper a aquest podria ser un bon indicatiu per a augmentar els recursos del sistema proporcionant paral·lelisme a l'hora de el servei de peticions a disc i tractar aixi de millorar el rendiment. En etapes posteriors del projecte seria interessant comprovar com afecta la quantitat de memòria física assignada a la nostre maquina al percentatge d'utilització. El següent resultat ha estat obtingut amb 2048 MB de memòria física.

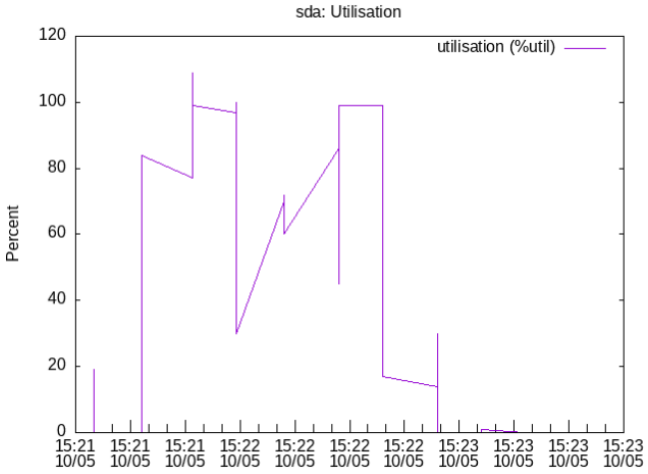

Fig. 19: %util del disc sda amb 2048 MB de memòria física

6.3.2 Peticions d'operacions d'E/S al dispositiu

La següent gràfica adjuntada a la figura 20 fa referencia a les mètriques relacionades amb les peticions de I/O al dispositiu, separades per escriptura i lectura. Les mètriques representades son r/s i w/s que identifiquen el numero de lectures i escriptures completades per segon. Aquest dues mètriques les hem combinat amb rrqm/s i wrqm/s que permeten veure la quantitat de peticions de cada tipus que s'han posat a la cua del dispositiu per a ser tractades. Fent un paral·lelisme amb el que succeeix a la aplicació podem veure que el numero de peticions que arriben a la cua es molt gran al inici del programa on es llegeixen moltes dades i es redueix considerablement a la resta de la execució. El ritme d'escriptures i lectures efectuades es de igual manera major al inici i es redueix gradualment a la resta de l'execució. Aquest subconjunt de mètriques no son gaire interessants a l'hora de prendre decisions respecte als recursos del sistema però si que permeten dimensionar l'aplicació amb la que es treballa i extraure informació de la manera com treballa respecte a la memòria.

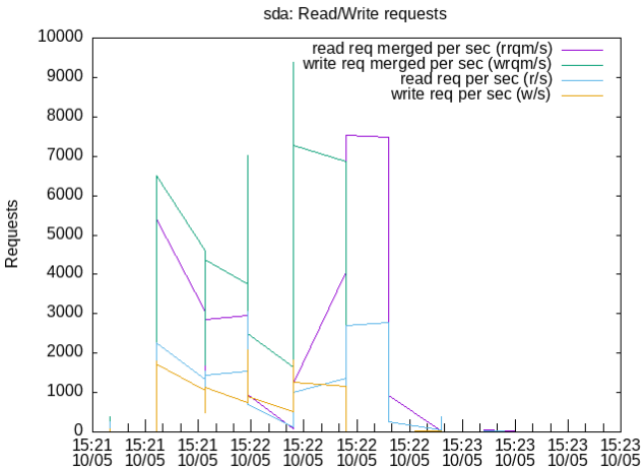

Fig. 20: Peticions d'E/S al dispositiu

6.3.3 Temps de servei de peticions i espera en cua

A continuació analitzarem mostrem les mètriques que fan referencia tant al temps de servei com el temps d'espera en cua de les peticions fins a ser servides, per una banda les de lectura a traves de r_await i w_await per a les escriptures. Aquestes mètriques son variables dependents de la mida de la cua, es a dir a major numero de peticions a disc, major serà el numero de peticions en cua i més temps esperaran a ser tractades. Degut a aixo es recomanable de cara a l'usuari que pretén fer un anàlisi de rendiment utilitzar el numero de peticions mostrades al dispositiu com hem vist a la figura 20 juntament amb la gràfica del temps d'espera en cua per tal de trobar correlació entre aquestes. Si observem ambdues gràfiques podem veure la relació en el ritme de peticions i el temps d'espera, a la primera gràfica podem veure que primer s'arriba al màxim de peticions d'escriptura i posteriorment de lectura i es veu exactament reflexat a la segona gràfica.

Com veiem a la figura 21 el pic de temps de servei coincideix amb el punt màxim de lectures empaquetades posades en cua i prèviament el pic de les escriptures.

Com recomana Mitchell Grande[8], qui també realitza un anàlisi del rendiment del disc, en termes generals podem considerar una latència de disc superior a 25 ms com a indicador de problemes de rendiment lleugers i superior a 50 ms com a problemes greus. A la nostra execució les lectures no causen gaire problema ja que en pocs casos superen els 10 ms i puntualment fan un pic de 50 ms, per altre banda les escriptures arriben a valors superiors a 30 en un punt especific del programa. Tenint en compte la execució complerta podem seguir concloent que el nostre sistema no es troba sota un clar cas de disminució del rendiment de manera globalitzada, però si hi han pics on degut a les peticions al dispositiu d'emmagatzematge les cues augmenten la mida en gran mesura de manera temporal.

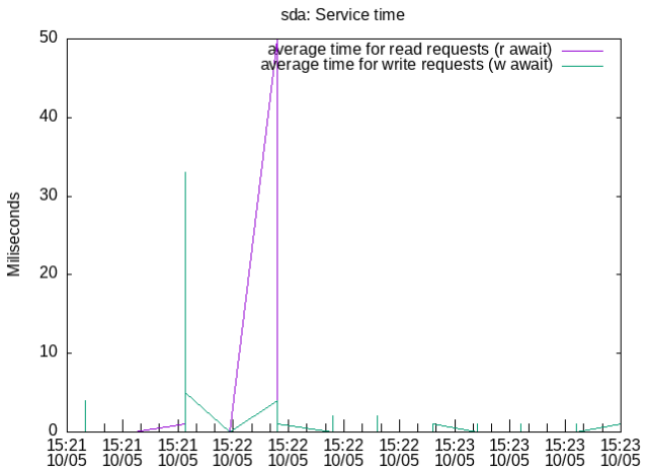

Fig. 21: Temps de servei de peticions i espera en cua amb 2048 MB

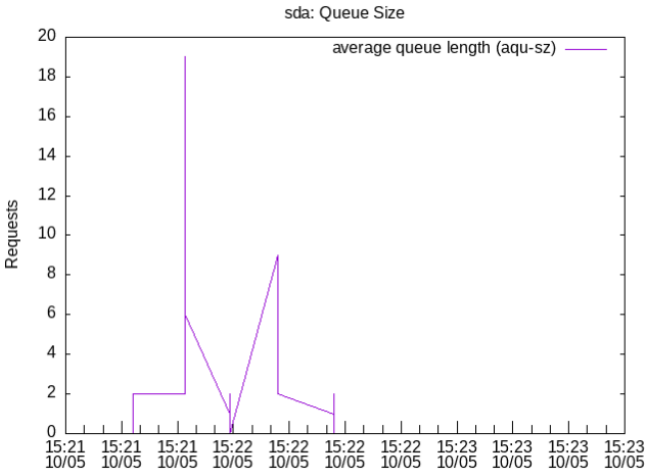

Fig. 22: Mida de la cua del *device* amb 2048MB

Aquesta mètrica ha sigut molt variable al llarg de les execucions depenent al ritme en que les peticions arribaven al disc i la quantitat d'aquestes que havien de ser posades en cua, degut a aixo s'han obtingut temps de servei instantanis molt alts, per sobre del que es considera un problema greu. Aquest comportament era puntual i no sostingut en la execució, en general totes les execucions seguien el mateix patró.

6.3.4 Mida de la cua

La mida de la cua del disc pot ser un indicatiu de possibles problemes en l'accés a disc. Aquesta cua augmenta i disminueix en funció de la quantitat de peticions. Es a dir si la cua es molt elevada de manera continuada vol dir que el ritme en que es serveixen les peticions no es suficient per a abastir les necessitats de la aplicació i el ritme en que aquesta realitza operacions d'E/S. Aquesta mètrica pot ser útil al usuari per a determinar si l'ample de banda del disc es el *bottleneck*.

En el nostre cas hem obtingut el resultat mostrat a la figura 22. Si comparem el que acabem, de descriure amb la figura en qüestió podem veure que en efecte, en la primera fase de la aplicació la cua augmenta degut a la gran quantitat de peticions en la lectura de les dades i la creació de la xarxa neural. A questa mida tenint en compte el numero de discs que serveixen les peticions en el nostre entorn es indicativa de que el dispositiu no treballa al ritme que l'aplicació li demana en aquest instant en concret però no hi ha gran problema a la resta de la execució. També hi ha un segon valor critic posteriorment que podem relacionar amb la lectura del fitxer de test. Si contrastem aquesta gràfica amb la de la figura 20 podem veure que la mida de la cua es molt gran en el moment inicial on el numero de peticions arriba al seu màxim.

La recomanació general que es fa es que la mida de la cua no sobrepassi el doble del numero de discs del teu entorn de manera sotinguda al llarg de l'execució. Per exemple si treballem amb un array de 5 discs una mida superior a 10 de manera continuada en el temps es indicativa de *bottleneck*.

6.3.5 Throughput

Per ultim analitzarem el thorugput que proporciona el disc per a comprovar l'ample de banda amb que ha servir les peticions. En aquest punt podem comparar les dades amb el test realitzat al apartat 6 del treball. Com podem veure en el nostre exemple hem arribat a 45.000 KB per segon en el moment de major activitat, aixo correspon a 45 MB/s, valor clarament inferior als 137 MB/s obtinguts al anàlisi preliminar, cal recalcar que aquests valors varien entre execucions i s'han arribat a màxims de 65 MB/s.

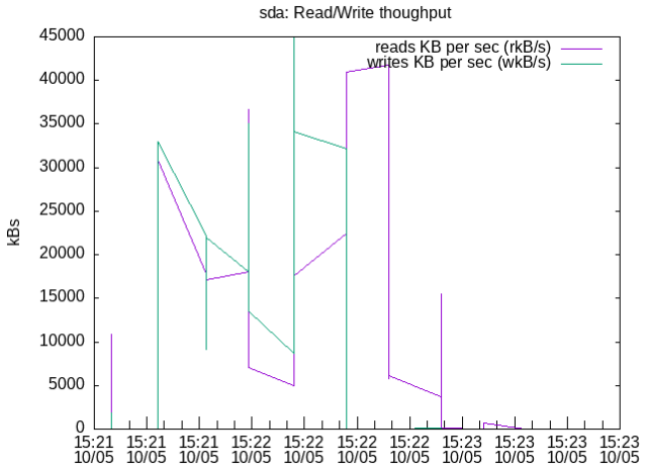

Fig. 23: Throughput amb 2048 MB

6.3.6 Swap i saturació de memòria física en el rendiment

Per a continuar l'anàlisi del rendiment del nostre sistema seria interessant contrastar les nostres dades del sistema de memòria utilitzant la eina iotop, aquesta com s'ha descrit en apartats anteriors proporciona informació a temps real del percentatge de temps que un proces esta realitzant o be *swap-in* a la zona d'intercanvi. A més a més podem comprovar si reduir la quantitat de memòria física assignada a més de augmentar el temps d'execució també fa augmentar el percentatge de *swap*.

En aquest punt utilitzem la comanda *sudo iotop -a* per a obtenir aquells processos que realitzen tasques de I/O i

mostrar els resultats de manera acumulativa al llarg del proces de mostreig. En utilitzar aquesta comanda amb 2048 MB de memòria física el percentatge de *swap* com veiem a la figura 24 es del 56% en *swap*.

```
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN     IO>    COMMAND
   51 be/4 root        0.00 B      0.00 B  0.00 % 27.60 % [kswapd0]
 1857 be/4 ivan        2.38 G      0.00 B 56.16 %  8.82 % python3 mnist.py
```

Fig. 24: Percentatge de *swap* en el proces de lectura de fitxers

Els temps d'execució obtinguts en execucions en fred, on la memòria principal no conte cap imatge i no conte les estructures pròpies de la xarxa neural degut a que es la primera vegada que executem la aplicació, son propers als 4 minuts(240 segons) de mitjana. Si realitzem altres execucions posteriors a la primera el temps es redueix així com les mètriques *DISK READ* i *swap-in* de iotop degut a que es redueixen les fallades en cache i la necessitat d'accés a disc.

A continuació reduirem el numero de recursos de memòria física, per a poder proporcionar al usuari un exemple de quins resultats es podrien esperar en cas de que el seu sistema sofreix de pressió de memòria greu, per sobre del experimentat amb 2048 MB, com afecta al rendiment, al percentatge de temps emprat en intercanvi de memòria i l'impacte en les mètriques mostrades gràficament.

6.4 Resultats amb 1700 MB de memòria física

L'objectiu d'aquest apartat era mostrar un cas de sistema amb una pressió de recursos molt elevada. En primera instancia aquest test estava planejat per a ser realitzat amb 1024 MB de memòria física. Finalment, els resultats que veurem a continuació han estat obtinguts amb 1700 MB de memòria. En l'apartat de discussió descriurem aquest canvi.

6.4.1 Utilització del dispositiu

En aquest exemple podem veure a la figura 25 que la quantitat de memòria assignada provoca una utilització elevada molt més constant que en el anterior exemple amb 2048 MB. Durant una gran fracció del temps el percentatge d'utilització es troba en el punt de saturació o molt proper a aquest. També podem notar que en alguns moments el percentatge mostrat es superior al 100.

Aquest fenomen es descriu en l'article de Peter Zaitsev[9] que fa un estudi similar al que hem fet nosaltres per amb multithreading, aixo provoca que el percentatge d'utilització sigui pràcticament al punt de saturació de manera estable, ja que almenys un thread esta utilitzant el disc. Peter ens descriu el següent: Aquest fenomen es degut a la manera en que la eina recull les dades. S'intenta fer mostreig de les dades cada segon, per en condicions de *overload* en el sistema el proces de recollida de dades pot trigar. Quan posteriorment es fan els càlculs pertinents per obtenir les dades apareixen màxims i mínims que no representen les dades reals. En termes generals poden ser ignorats.

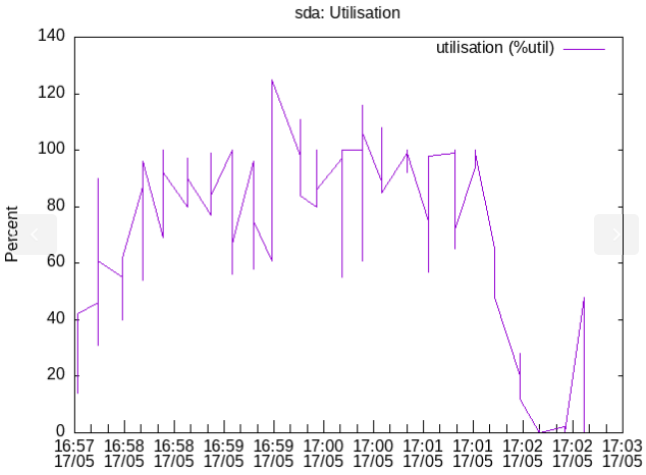

Fig. 25: %util amb 1700 MB de memòria

6.4.2 Peticions d'operacions d'E/S al dispositiu

Com podem veure en la figura 26 el numero de peticions realment efectuades a disc tant escriptures com lectures no difereix tant com podríem esperar en termes globals de l'exemple anterior, aquesta es la conclusió que podem treure si únicament ens fixem en les lectures efectuades per segon(r/s) i en les escriptures efectuades per segon(w/s).

En canvi si ens fixem en les mètriques rrqm/s i wrqm/s relacionades amb la quantitat de peticions de les que s'ha fet *merge* per segon, podem veure que son molt més elevades en els punts crítics i prolongades en el temps, degut a que la execució es més llarga. Podem concloure que les operacions I/O efectuades son de major mida, ja que en ser posicions de memòria consecutives podem aprofitar aquestes peticions i realitzar-les com una sola més gran. Aquest fenomen el podem comprovar observant les mètriques *rareq-sz* i *wareq-sz* de ambdues execucions, aquestes mètriques ens indiquen la mida en kb de les peticions, i es pot observar una clara diferencia.

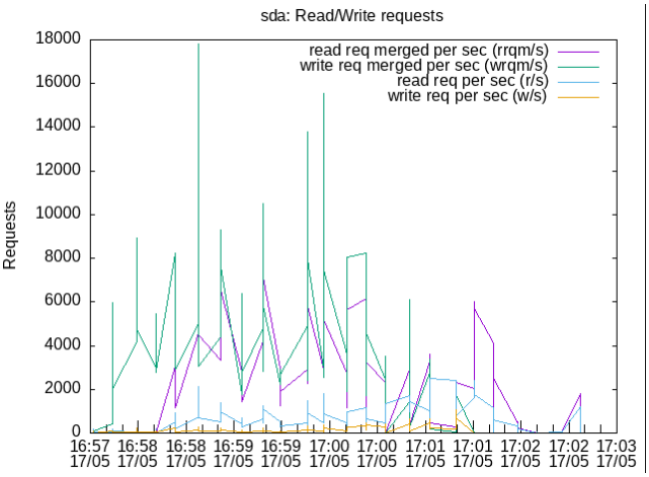

Fig. 26: peticions d'E/S

6.4.3 Temps de servei de peticions i espera en cua

A la figura 27 podem observar que els resultats es troben clarament per sobre del que es consideraria greu. Si

comparem aquestes dades amb les obtingudes en la latència de l'anàlisi preliminar podem veure que la diferencia es molt gran. Podem esperar que la mida de la cua sigui gran en la majoria de temps degut a que el disc no proporciona les dades al ritme en que la aplicació les demana, aquest fenomen es una mostra de *bottleneck* per iops.

En el cas de 2048 MB el màxim era proper als 100 ms però no era sostingut al temps, en aquest exemple, obtenim valors superiors a aquest i de manera continuada, sobretot en el cas de les escriptures.

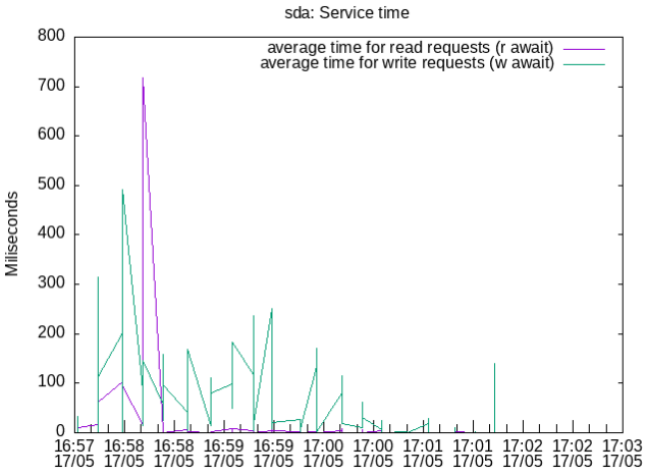

Fig. 27: temps de servei amb 1700 MB de memòria

6.4.4 Mida de la cua

El que observem a la mida de la cua ve molt relacionat amb el que comentàvem amb el temps de servei. Els primers moments de las aplicació i les crides intensives al disc provoquen que la cua s'empleni amb una quantitat de peticions elevada tenint en compte que les peticions es donen servei de manera seqüencial. El dispositiu te un temps de servei molt elevat degut a que no pot servir aquestes peticions. Aquesta mètrica ens indica que la figura 27 mostra el temps de servei i de espera en cua, però que realment en aquest cas el factor determinant es el percentatge d'aquest temps que esta en cua i no en que es serveix la pròpia petició.

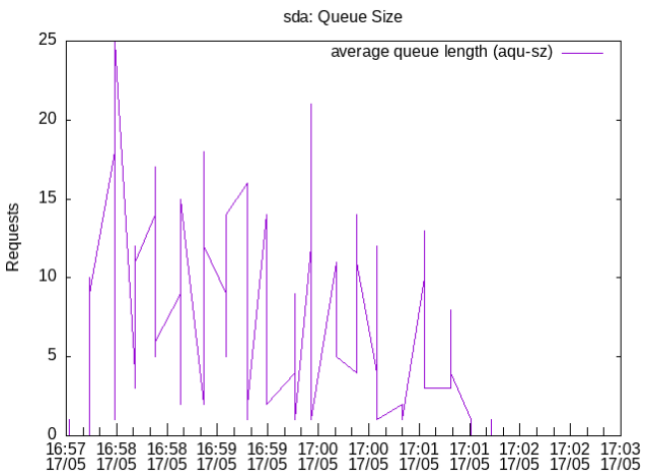

Fig. 28: mida de la cua amb 1700 MB de memòria

6.4.5 Throughput

De igual manera que en la execució anterior no veiem que el numero de lectures i escriptures en termes generals siguin molt elevats excepte en l'inici de la segona meitat de l'execució on s'arriba a 120 MB/sen primer lloc i posteriorment arriba a un màxim de 190 MB/s. Aixo ho podem relacionar amb la reducció de les crides intensives, i la disminució de la mida de la cua, en termes generals, el dispositiu deixa d'estar tant saturat en aquesta etapa. En aquest punt i si ho comparem amb la figura 8 podem veure que s'arriba de manera raonable al nivell de *througput* obtingut en l'anàlisi preliminar.

Quan parlem del *througput* hem de tenir en compte que encara que el valor ideal sigui el que hem obtingut en l'anàlisi preliminar de prestacions, o en molts casos del mon real, el que el propi fabricant ens ofereix en les especificacions del dispositiu, hem d'esperar que en entorns de producció difíclmemt obtindrem aquests valors ja que el disc es veu afectat per molts factors que poden ser limitatnts del seu *throughput*.

Per a entendre el que succeeix en aquesta gràfica podem complementar la informació que ens ofereix amb la corresponent gràfica del temps de servei(r_await i w_await). Al inici de l'aplicació com ja sabem hi ha un període de crides intensives, aixo provoca un augment dràstic en el *workload* del HDD i un augment del temps de servei considerable, degut a que el temps d'espera en cua augmenta com hem explicat anteriorment. No es fins que el nivell de saturació baixa i el temps de servei torna a valors més petits quan el *throughput* del dispositiu arriba als màxims obtinguts. En altres paraules, augmentar la concurrència del servei d'aquestes peticions podria millorar considerablement el rendiment en les etapes d'entrenament i test.

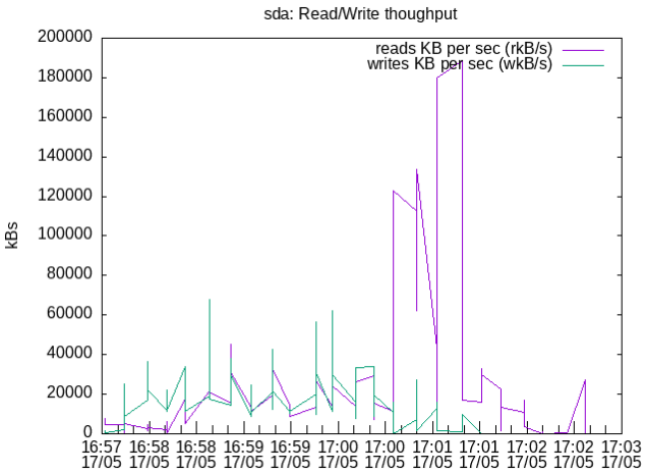

Fig. 29: throughput amb 1700 MB de memòria

6.4.6 Swap i saturació de memòria física en el rendiment

Amb una quantitat de 1700 MB de memòria física podem veure que el percentatge de *swap* s'ha elevat fins al 61% en l'exemple de la figura 30, encara que al llarg de les execucions realitzades hem obtingut un 65% de mitjana. També hem de tenir en compte que el proces encarregat en unix de

gestionar la memòria virtual (kswapd0) te un percentatge d'espera major en operacions de I/0, passant d'un 25% en el exemple amb 2048 MB a 35% en aquest cas. També podem observar que el numero de lectures a disc ha augmentat considerablement on nomes es llegien 2.48 GB en la versió amb 2048 MB.

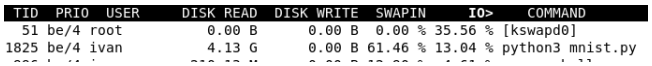

Fig. 30: percentatge de *swap*

6.5 Memòria inferior a 1700

Com s'ha comentat en apartats anteriors d'aquest informe, el nostre pla inicial en el testeig amb diferent quantitat de recursos era utilitzar tres quantitats de memòria molt diferents que representessin tres sistemes molt diferents pel que fa a l'us de recursos. El primer d'ells un cas amb 4096 MB, on esperàvem no tenir problemes i poder observar el comportament de la aplicació i quin ritme de treball demana al disc, el segon amb 2048 MB on esperàvem trobar problemes lleugers, i un ultim amb 1024 on tenim un sistema de emmagatzematge completament saturat.

Finalment hem hagut de realitzar uns canvis i l'últim els tests ha sigut amb 1700 MB de memòria física. En aquest apartat tenim com a objectiu descriure el motiu i quines opcions te l'usuari per a detectar la problemàtica que descriurem a continuació.

El nostre sistema de emmagatzematge principal conte aquells programes que son utilitzats majorment juntament amb el sistema operatiu. Si la memòria es molt reduïda, amb els sistemes operatius actuals cal reservar una quantitat de RAM al funcionament del propi OS relativament elevada. En el cas de debian 11.2, que es el que nosaltres utilitzem, requereix una quantitat mínima de 512 MB i una recomanada de 2GB. Si tenim en compte que ne el nostre cas l'aplicació executada es una xarxa neural, que solen ser programes que ocupen gran quantitat de recursos e memòria, ja que contenen estructures de dades grans per a representar les capes de la xarxa, es possible que unix faci *kill* del process per a garantir el correcte funcionament del sistema operatiu.Si el OS no te espai en RAM i s'ha de fer paginació en la zona d'intercanvi de processos del propi OS, aquest podria disminuir molt les seves prestacions i posar en perill el sistema.

Per a comprovar que efectivament el teu proces ha estat eliminat per aquestes raons, ja sigui una xarxa neural o un altre aplicació amb crides intensives a memòria, podem accedir si tenim permisos de administrador en el ficher /var/log/syslog. Aquest fitxer proporciona infirmació sobre les peticions dels processos a memòria, també informa de que s'ha fet *kill* del proces en qüestió i el motiu s'indica com *Out of memory.* El conegut OOM killer es el sistema de linux encarregat de quan el sistema esta apunt de quedarse sense memòria, matar els processos corresponents.

6.6 Discussió i interpretació dels resultats

En aquest apartat en disposem a discutir els resultats obtinguts i quins aspectes ens semblen més importants i interessants a tenir en compte a l'hora de realitzar aquest tipus d'estudi.

6.6.1 Límit inferior de recursos

Comencem per un concepte amb relació amb el que hem explicat en el subapartat anterior. Quan treballem amb la nostre aplicació podem calcular la quantitat mínima de recursos per no generar *swap* fent servir eines de monitorització. En un entorn real no volem en cap cas provocar *swap* en executar aquest tipus d'aplicacions però si que podem degut a interessos econòmics minimitzar l'us dels recursos i no demanar de més. En tot cas pot ser que per interessos econòmics vulguem calcular la quantitat mínima de recursos que ens calen, el més interessant seria calcular aquesta quantitat i afegir un marge raonable per a no condicionar el sistema en cas d'un pic inesperat d'us de recursos. En termes generals els recursos necessaris per a no generar *swap* el podem calcular observant l'us de memòria física per part del sistema en repòs, per el propi OS, i l'us de la aplicació en el seu punt màxim d'us de recursos.

Per a dur a terme aquest calcul podem utilitzar l'eina *top*, aquesta permet visualitzar l'us de memòria física global del sistema i també desglossat per cada proces. En primer lloc observem l'us en repòs de memòria física, a la següent figura podem observar la quantitat de memòria física utilitzada en repòs. Com podem veure es troba proper als 700 MB, concretament 182 MB per sobre del mínim requerit per a suportar aquesta distribució Linux, aquest valor varia lleugerament segons els processos que estan executant-se en segon pla.

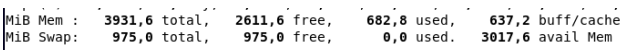

Fig. 31: Memòria física en repòs

Seguidament executem el nostre programa per a observar el pic d'us de memòria, es a dir aquella memòria mínima que no provoca *swap*. A la figura 32 podem veure que el us màxim de recursos es troba lleugerament inferior a 2500 MB. A llarg de totes les proves realitzades en cap cas ha sobrepassat aquest valor, per tant podríem considerar la frontera entre la execució pròpiament de la aplicació, i la execució amb intrusió dels processos de *swap* en disc es troba en aquest limit. En el requadre mostrat podem veure l'us de memòria en el punt màxim.

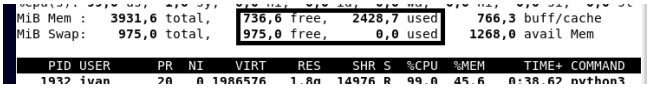

Fig. 32: Memòria física amb execució de la aplicació

6.6.2 *Bottleneck* segons el numero de recursos

Les xarxes neurals son intensives en comput, generalment i en condicions normals, aquest es el factor limitant de

l'execució. En el nostre estudi hem fet un primer test amb una quantitat de memòria elevada, on no es produeix la necessitat de fer *swap* de processos a disc. En aquest primer exemple la CPU es el factor limitant de la execució. Fent ús de la eina top s'ha observat un us elevat de CPU i un percentatge d'us de la RAM inferior al 100%, ja que no saturem el recurs, per tant el disc no suposa un *bottleneck* ja que nomes es llegeix de disc la carrega d'imatges. Es pot comprovar ràpidament si comparem els temps d'execució amb un valor de memòria suficientment per sobre del limit calulat com per a no provocar *swap* amb els temps obtinguts amb 4096 MB, en ambdós casos obtenim els mateixos temps d'execució.

Quan reduïm els recursos per sota lleugerament del limit inferior estimat, provoquem *swap* i aixo clarament es veu afectat en els temps d'execució. El *swap* provoca crides al sistema d'emmagatzemtage secundari i per tant el rendiment esta limitat per iops en gran part de l'execució.

6.6.3 Augment de les lectures i escriptures a disk

Com hem pogut veure a les gràfiques representades i la informació que ens proporciona la mètrica DISKREAD de iotop hem vist que la quantitat de memòria llegida de disc augmenta en funció de la memòria física assignada i per tant de la quantitat de *swap* necessària. En aquest apartat veurem la correlació entre el *swap* i la quantitat de memòria llegida de disc.

Degut als processos de *back-propagation* i *forward-propagation* de la xarxa neural aquesta ha de accedir a les seves estructures de dades repetidament al llarg del programa oper a realitzar operacions i actualitzar els pesos de les capes per a reduir l'error del model. Si la memòria es limitada i les estructures de dades no ens caben en memòria llavors es genera una gran quantitat de *trashing*, per tant a menor quantitat de RAM més accessos a disc per a poder dur a terme el proces de *cross-validation*.

TAULA 1: DADES LLEGIDES I SWAP

RAM	DISK READ	SWAP
1700 MB	4,13 GB	65%
2048 MB	2,38 GB	56%
4096 MB	153 MB	0%

Com hem vist al llarg de tots els exemples, l'augment d'us de memòria virtual esta directament relacionat amb la saturació de la memòria física. Per a saber quina memòria disponible ens falta per a arribar a aquest punt de saturació, l'usuari pot utilitzar l'eina top, concretament la mètrica *avail Mem* per veure a temps real la quantitat de memòria que pot ser alocatada pels processos abans de provocar més *swap*. Aquestes escriptures al disc provoquen un endarreriment en la execucio del programa, la CPU esta en espera més temps degut a aquestes operacions.

6.6.4 CPU i SWAP: temps d'espera per I/0

Podem concloure que hi ha una relació directa entre la quantitat de *swap* requerida i el temps d'espera de la CPU. Per a observar aquest fenomen cal tenir en consideració la mètrica que ofereix *iostat* en el apartat de informació de CPU, aquesta mètrica s'anomena *%iowait* i fa referencia al percentatge de temps que la nostre CPU ha estat esperant degut as que s'estan produint crides al disc per a obtenir dades necessàries per el programa. A la taula 3 podem observar com en augmentar el *swap* també ho ha fet el temps en que la CPU ha estat en espera i no ha pogut per tant fer altres operacions. Aquest fet deixa en evidencia el concepte de execució limitada per iops que hem comentat en l'apartat anterior. En la taula es mostra la mitja del percentatge, s'ha de tenir en compte que en el cor del programa, on les peticions a disc son majors, gairebé el percentatge no baixa del 75%, estant proper a 90% la majoria del temps en el cas de la execució amb 1700 MB. De igual manera passa a la resta d'exemples, però amb menor impacte ja que els valors del percentatge de temps en espera no defereixen tant entre les etapes de crides intensives i la resta del programa

La CPU esta relacionada amb el *swap* degut a que a major quantitat de *swap-space* requerit, major es la quantitat de decisions que ha de prendre la CPU sobre quins processos i on s'ha de fer l'intercanvi entre memòria virtual i física. Però si el nostre disc no es lo suficientment ràpid, la CPU veurà afectat el seu funcionament degut a les esperes experimentades i a que hem descrit en aquest apartat.

TAULA 2: %IOWAIT EN FUCIO DE SWAP

RAM	SWAP	%iowait
1700 MB	65%	48'28%
2048 MB	56%	15,35%
4096 MB	0%	1,06%

6.6.5 Temps d'execució

El temps d'execució del programa ve relacionat amb l'us de la memòria virtual i la necessitat d'accés a disc(zona d'intercanvi), a mesura que s'han de fer un major numero de moviments de dades entre disc i memòria principal augmenta el temps d'execució. A la taula adjuntada podem veure les dades obtingudes amb l'eina *time* per a calcular el temps d'execució. i la relació directa entre la memòria assignada i el resultat en qüestió. Per a valors de memòria per sota del limit inferior calculat per a aquest programa els temps augmentaven dràsticament, arribant a obtenir temps d'execució de fins a 20 minuts abans de que el sistema operatiu mates el procés.

TAULA 3: TEMPS D'EXECUCIÓ

RAM	Mitjana temps d'execució
1700 MB	329 s
2048 MB	240 s
4096 MB	111 s

7 CONCLUSIONS

En aquest treball hem fet un estudi de les eines de monitorització que ens ofereix el sistema per a fer un anàlisi de prestacions del nostre disc en funció del nombre de recursos. S'han descrit quines mètriques poden ser útils, com podem fer un mostreig eficient d'aquestes i com es poden mostrar al usuari gràficament per a la presa de decisions. Aquest treball pot servir de guia a qualsevol usuari que treballi amb aquesta mena de aplicacions de IA per tal de fer un estudi de prestacions i treure conclusions sobre el funcionament del seu sistema, detectar ineficiencies i *bottlenecks*.

AGRAÏMENTS

El més sincer agraïment a la meva tutora Dolores Rexachs per la seva guia i ajut al llarg del meu treball de final de grau.

REFERÈNCIES

[1] Thakur, Rajeev Lusk, Ewing. (2000). I/O in Parallel Applications: The Weakest Link. International Journal of High Performance Computing Applications. 12. 10.1177/109434209801200401.

[2] Levine, Gropp and Galbreath, "Applications-driven parallel I/O,ïn SC Conference, Portland, OR, USA, 1993 pp. 462-471. doi: 10.1109/SUPERC.1993.1263494

[3] https://medium.com/naukri-engineering/understanding-disk-i-o-when-should-you-be-worried-naukri-engineering-f0ab332f52d4

[4] https://man7.org/linux/man-pages/man1/iostat.1.html

[5] http://blog.josemarianoalvarez.com/2018/06/10/el-perceptron-como-neurona-artificial/

[6] https://github.com/markcurtis1970/graph-iostats

[7] https://hmong.es/wiki/Memory_access_pattern

[8] https://www.concurrency.com/blog/september-2019/diagnosing-disk-performance-issues

[9] https://www.percona.com/blog/2017/08/28/looking-disk-utilization-and-saturation/

[10] https://goughlui.com/the-hard-disk-corner/hard-drive-performance-over-the-years/

APÈNDIX

A.1 Mètriques de les eines de monitorització

A.1.1 iotop

- DISK READ: ample de banda de lectura a disc durant el proces de mostreig.
- DISK WRITE: ample de banda de lectura a disc.
- SWAPIN: percentatge de cada proces emprat en *swap*.
- IO: ens mostra el percentatge del temps de cada proces en tasques d'E/S.

A.1.2 iostat

- iowait: mostra el percentatge de temps que la CPU o CPUs estaven esperant per operacions de E/S.
- Device: indica el nom dels dispositius tal i com estan definits en el directori /dev.
- tps: indica el numero de transferències per segon que s'han realitzat al *device*. Entenem per transferència com una petició d'E/S.
- Kb_read/s: expressa la quantitat de dades llegides en el *device* en kb/s.
- Kb_wrtn/s: expressa la quantitat de dades escrites en el *device* en kb/s.

Si afegim a la nostre comanda la opció -x obtenim un output mes complert amb estadístiques de disc extres.

- r/s: nombre de peticions completades per segon.
- rrqm/s: numero de solicituds de lectura *merged* que es van posar a la cua del dispositiu.
- r_await: temps mig(milisegons) de les peticions de lectura que arriben al *device*. Aquest temps inclou tant el temps en cua com el temps emprat en servir la petició.
- %util: percentatge d'utilització del dispositiu.