Anàlisi de prestacions de sistemes d'emmagatzematge per IA

Joan Murciano Soto

Resum– Els programes d'Intel·ligència Artificial (IA) són programes que fan moltes lectures de fitxers per la seva naturalesa. Aquestes lectures requereixen moltes crides a dispositius d'emmagatzematge, i aquestes poden comportar endarreriments en l'execució del programa. L'ample de banda per transportar dades de disc a memòria o viceversa pot esdevenir en un *bottleneck*, incrementant el temps d'execució. De manera que és important saber detectar en aquest tipus de programes, si les entrades/sortides (E/S) del nostre sistema se saturen. En aquest treball s'estudien diferents programes amb altes quantitats de lectures a disc. S'utilitzen eines de monitorització, les quals ens informen amb mètriques relacionades amb E/S a disc. També veiem l'impacte que té el *swap*, el qual també provoca un increment d'operacions d'E/S. Aquest document pretén mostrar la metodologia utilitzada per a realitzar l'anàlisi descrivint les eines i els resultats obtinguts amb l'objectiu de que serveixi de guia per a entendre el comportament i l'efecte de les E/S i el *swap*.

Paraules clau– E/S, swap, IA, monitorització.

Abstract– Artificial Intelligence (IA) programs make many file readings by nature. These readings require many calls to storage devices, and this can lead to program execution delays. The bandwidth for transporting data from the device to memory or vice versa can become a *bottleneck*, increasing the runtime. It is important, in this type of software, to be able to detect if disk I/O is saturated. In this work, different programs with high amounts of disk reads are studied using monitoring tools that inform us about I/O-related metrics. We will also see the impact of *swap*, which increases I/O operations. This document aims to show the methodology used to perform the analysis by describing the tools and results obtained with the aim of serving as a guide to understanding the behavior and effect of I/O and *swap*.

Keywords– I/O, swap, AI, monitoring.

1 Introducció - Context del treball

La Intel·ligència Artificial (IA) és l'habilitat dels ordinadors per a fer activitats que normalment requereixen intel·ligència humana, utilitzant algoritmes, aprendre de les dades i utilitzant allò après en la presa de decisions[1]. El mode en què la IA és capaç d'obtenir aquestes capacitats és gràcies a l'aprenentatge automàtic (Machine Learning). N'hi ha 3 tipus: l'aprenentatge supervisat, el qual utilitza dades prèviament etiquetades amb una categoria; el no supervisat, aquí els propis algoritmes han de categoritzar les dades obtingudes; i l'aprenentatge per reforç, on els algoritmes aprenen de la seva pròpia experiència[1]. A mida que ha avançat la tecnologia, el BigData ha generat una gran demanda de sistemes amb una intel·ligència avançada. Existeixen moltes plataformes actualment que ens ofereixen aquests serveis, com ara Azure Machine Learning, Amazon Machine Learning, TensorFlow o BigML.

Dins el Machine Learning, hi ha un sots camp anomenat Deep Learning. Aquest utilitza una estructura jeràrquica de xarxes neuronals artificials, que es construeixen d'una forma similar a l'estructura neuronal del cervell humà [2]. Per a poder realitzar aquesta ingent quantitat de càlculs és idoni l'ús de GPUs (Unitats de Processament Gràfic), de la mateixa manera serà més eficient treballar amb aquestes aplicacions en entorns distribuïts, d'aquesta manera reduint en gran mesura el seu temps d'execució. Però ens trobem amb un problema, els frameworks més utilitzats (com TensorFlow) moltes vegades tenen mancances pel que fa a monitoritzar certes mètriques de cara al desenvolupador d'aplicacions,

• E-mail de contacte: joan.murciano.soto@gmail.com
• Menció realitzada: Enginyeria de Computadors
• Treball tutoritzat per: Dolores Isabel Rexachs Del Rosario (Departament d'Arquitectura de Computadors i Sistemes Operatius)
• Curs 2020/2021

com ara el rendiment d'E/S[3]. Això comporta no poder utilitzar tècniques d'optimització adequades. De manera que si volem optimitzar el rendiment de l'aplicació, haurem d'obtenir certes dades amb altres eines de monitorització. Aquí és on centrem l'atenció del nostre estudi.

1.1 Dispositius d'emmagatzematge i E/S

Quan parlem de les E/S, parlem de les crides a dispositius d'emmagatzematge que la CPU realitza per transportar dades carregades en memòria principal fins al dispositiu, o per carregar en memòria principal dades que es troben al dispositiu. És molt habitual en els programes d'IA trobar-se amb moltes operacions d'E/S, ja que han de fer moltes lectures de fitxers. Actualment, els discs durs són els dispositius d'emmagatzematge secundari més utilitzats[4], sent el primari la memòria RAM. I els discs, determinen en gran mesura el rendiment global del sistema, en concret tenen molt efecte a les aplicacions en les que es realitzen moltes operacions d'E/S[4]. Mentres que en altres components com la CPU o la RAM han augmentat el seu rendiment i ample de banda a mida que ha avançat la tecnologia, els discs ho han fet en menor mesura tot i ser components fonamentals en el sistema d'emmagatzematge. Segons Pérez[5], va haver un canvi de mentalitat a l'evolució de la informàtica. Entre els anys 60 i 80 va ser l'època de la revolució de la computació, però a partir dels 90 va aparèixer l'època de la informació, donant més importància als sistemes d'emmagatzematge. Tot i així, els discs no han avançat a un ritme adecuat[5]. És a dir, l'ample de banda ofert pel disc d'E/S acaba identificant-se com un dels majors *bottlenecks* del rendiment a l'hora d'executar un programa d'IA. Una possible solució és fer ús de sistemes distribuits amb sistemes d'E/S en paral·lel, però les E/S continuen sent un dels principals *bottlenecks*, també a causa del desequilibri entre el temps de còmput i d'E/S[5].

Cal esmentar que hi ha diferents tipus de dispositius d'emmagatzematge secundari. Els més utilitzats són els disc durs (HDD) i les unitats d'estat sòlid (SDD). Gonzalez[6], fa una comparativa entre aquests dos tipus de dispositius a la seva tesis amb la qual demostra que els SSD tenen una velocitat de transferència molt més elevada, comportant un millor rendiment en l'execució de qualsevol programa en la majoria dels casos. Un SSD té els següents avantatges davant d'un HDD: És més ràpid, te un consum d'energia inferior, menys susceptibilitat a la pèrdua de dades, menys soroll durant el seu funcionament i més compacte. Però te dos principals problemes: el preu es molt més elevat i la vida útil. La durabilitat de les seves cèl·lules de memòria ve definida pel nombre d'operacions d'escriptura realitzades[7]. Aquestes són les principals raons per les que encara s'utilitzen els discs durs, tot i no ser l'opció més eficient.

1.2 Swap i memòria virtual

Hi ha casos en els que un sistema no compta amb la memòria física (RAM) necessària per a carregar totes les dades que necessita un programa en execució. Quan hi ha processos que necessiten més memòria física de la disponible, es fa ús de la memòria virtual, i es genera swap (o intercanvi de memòria)[8]. Aquest és un espai d'intercanvi

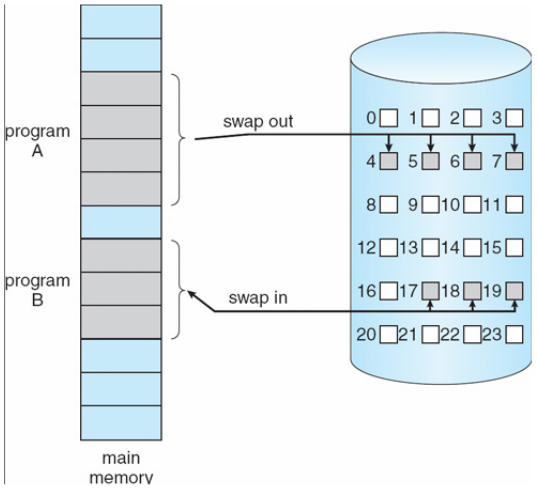

Fig. 1: Transferència en swap de dos programes.[11]

de memòria que ve limitat per la quantitat d'espai d'emmagatzematge disponible (quant més espai, per exemple, de disc dur, més gran pot ser l'espai de disc utilitzat per a *swap*). S'utilitza aquest espai per emmagatzemar dades temporalment a falta d'espai a la memòria principal[8]. Una de les tècniques de memòria virtual més utilitzades és la paginació[9]. En aquesta, la memòria principal es divideix en marcs i cada procés es divideix en pàgines del mateix tamany que els marcs, i aquestes pàgines es carreguen en memòria en marcs lliures[10]. Quan es necessiten les dades d'una pàgina que es troba al dispositiu d'emmagatzematge (zona anomenada *swap space*) es provoca un *error de pàgina*, es fa *swap-out* i es porta una pàgina carregada en memòria principal que es detecti que no està sent gaire utilitzada al *swap space*, per després fer *swap-in* i portar la pàgina necessària del *swap space* a la memòria principal[10]. Hi ha molts algoritmes que poden ser aplicats a aquesta tècnica, i poden afectar significativament al rendiment del programa. A la figura 1 veiem un exemple de l'esmentat anteriorment.

Hem de tenir en compte un aspecte molt important, i és que cada cop que es fa una crida a un dispositiu per a fer operacions *swap*, les hem de concebre com operacions d'E/S. De manera que cada cop que s'accedeixi al *swap space* d'algun dispositiu per a realitzar intercanvi de memòria, el rendiment del programa pot resultar afectat per les raons d'ample de banda i latència abans esmentades[12]. Sempre que s'intenti accedir a una zona d'intercanvi, costarà més lent que un accés directe a la memòria RAM[12].

1.3 Eines de monitorització

A la figura 2 observem diferents eines de monitorització de Linux, desglossat en les capes en les quals observen. Per a fer el nostre estudi utilitzem iostat, iotop, top, vmstat i perfstat i Darshan. Les eines escollides odfereixen mètriques relacionades amb E/S i *swap*, més endavant s'explica el perquè s'ha triat cadascuna d'elles.

Aquests possibles problemes relacionats amb les E/S, poden suposar un malbaratament energètic, temporal i material. De manera que ens proposem monitoritzar diferents programes i analitzar les mètriques extretes, creant un in-

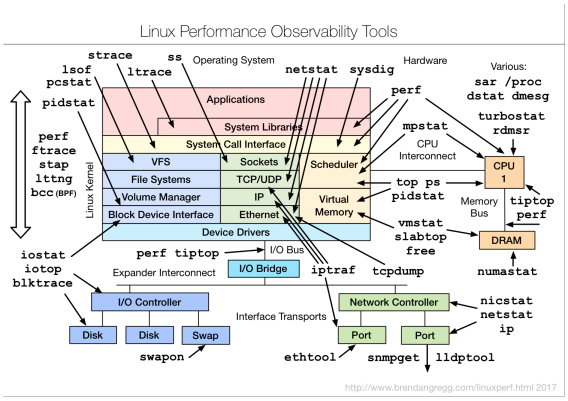

Fig. 2: Linux Performance Observability Tools[13]

forme en el que es descriu el seu comportament i així ajudar a trobar possibles *bottlenecks* relacionats amb les E/S a una aplicació. L'organització de la resta del document està separat en els següents apartats: Objectiu global i objectius específics; Metodologia utilitzada, planificació i les etapes d'investigació realitzades; Explicació de les eines de monitorització utilitzades; Resultats; Discussió; Conclusions; Bibliografia; Una secció a manera d'apèndix.

2 Objectiu Global

L'objectiu d'aquest estudi té com a finalitat fer una anàlisi del comportament de les entrades/sortides (E/S) d'una aplicació d'IA, i analitzar el seu rendiment i temps d'execució.

2.1 Objectius específics

Per assolir aquest objectiu global, el dividim en diferents objectius més específics:

1. Entendre el funcionament i les mètriques de diferents eines de monitorització que ofereixen informació relacionada amb E/S i *swap*.

2. Analitzar l'execució de diferents aplicacions a petita escala amb un còmput molt baix, amb finalitat de poder transportar la metodologia a un còmput superior.

3. Comprendre els canvis que pot patir el temps d'execució d'aplicacions amb alta quantitat d'operacions d'E/S segons els recursos que se li administrin al sistema en el qual estan sent executats.

4. Crear una documentació amigable de totes les mètriques mesurades amb la finalitat de comprendre millor les E/S que es produeixen a les aplicacions i com afecta al seu rendiment.

3 Metodologia

Per a portar a terme el nostre estudi, utilitzem diferents eines d'observabilitat del rendiment. Aquestes ens permeten monitoritzar tot tipus de moviment d'informació que ocorre a una màquina i els processos que hi participen, observant entre altres mètriques relacionades, les E/S. Per a triar les eines a utilitzar, s'ha investigat i fet ús de més eines de les que apareixen en aquest document, però algunes d'elles han estat descartades. Les eines finalment utilitzades i que es veuen reflectides als resultats són les següents: Iotop, Top, Iostat, Vmstat, Perf-stat i Darshan. També s'han utilitzat dos mòduls d'optimització del kernel Linux que milloren el rendiment, evitant la paginació a disc. En un sotsapartat més endavant s'explica detalladament cada eina. El projecte s'ha portat a terme amb un model en cascada. Cadascuna de les etapes, s'han anat definint a mida que ha avançat el projecte. Segons els resultats obtinguts a cadascuna de les etapes, s'ha concretat els detalls per a seguir amb la següent etapa.

3.1 Entorn de treball

Per a fer el nostre estudi, utilitzem màquines virtuals amb el software Oracle VM VirtualBox. El sistema operatiu és Debian 10.8. La raó d'utilitzar una distribució Linux, és que te eines més específiques amb obtenció de les mètriques que ens interessa analitzar, amb fàcil obtenció i ús d'aquestes. A l'hora, ens resulta més fàcil i ràpid arrancar i apagar màquines virtuals amb distribució Linux. Quelcom necessari en el nostre estudi per a poder administrar a la màquina diferents quantitats de memòria física i comparar els resultats obtinguts. Les quantitats administrades de memòria es troben en el rang de 800 MB fins a 3500 MB (depenent del programa a analitzar). I hem administrat 50 GB d'emmagatzematge de disc dur (HDD). El fet d'utilitzar un HDD enlloc de SSD és perque els HDD busquen i recuperen dades amb més lentitud. Això ens ajudarà a poder identificar problemes d'E/S més fàcilment.

3.2 Planificació

Com hem comentat, el model que s'ha utilitzat ha estat en cascada, afectant de la mateixa manera a la planificació. Cadascuna de les etapes de la planificació coincideix amb les etapes d'investigació que s'expliquen al següent apartat. Per a veure gràficament i amb més detall, observar el Diagrama de Gantt que es troba a l'apèndix.

3.3 Etapes d'investigació

En aquest apartat s'exposa cadascuna de les etapes d'investigació que hi ha hagut fins a la finalització del treball. Ho separem en 3 apartats en els que els objectius a realitzar evolucionen, a causa del coneixement adquirit a cada etapa.

3.3.1 Anàlisi de programa de crides intensives d'E/S

Prèviament a aquesta etapa, fem ús de l'eina "strace"per a comprovar que amb un senzill programa d'escriptura en un fitxer estem fent crides a disc d'E/S d'escriptura, d'aquesta manera assegurant-nos que realment s'estan fent aquestes crides. El programa està escrit en python i es fan escriptures en un fitxer. Es comencen a conèixer les diferents eines esmentades anteriorment a analitzar les dades obtingudes. Per a no estar creant fixers de grans dimensions i ocupar espai al disc, es tria crear un fitxer de 4GB amb l'anterior programa d'escriptura. A partir d'aquest moment, es comencen a monitoritzar possibles *bottlenecks* d'E/S quan es fan accessos a aquest fitxer, tant en aquesta etapa com en la següent. En aquesta primera etapa executem un programa que anomenem "read3"per a realitzar la lectura d'aquest

fitxer. Es monitoritza aquest programa amb la idea de trobar mètriques que ens demostrin aquest *bottleneck*, ja que aquest programa no fa res més que lectures (39.99 MiB).

3.3.2 Anàlisi de programa d'E/S a causa del *swap*

El programa està escrit en python i es fan lectures d'un fitxer (3.90 MiB). Veiem a l'anterior etapa que al modificar la quantitat de memòria física administrada a la màquina virtual, es pot veure afectat el rendiment. O vist des d'una altra perspectiva, si modifiquem la quantitat de dades a carregar a memòria. En aquesta etapa ens fixem en que el *swap* provoca problemes d'E/S, ja que s'ha de fer ús de la memòria virtual i emmagatzemar en disc dades que haurien d'estar en memòria física, però aquesta és insuficient. Veient aquests problemes, es crea un programa que anomenem "read4", el qual realitza el mateix nombre de crides d'E/S que a l'anterior programa read3, però estructurant el codi de tal manera que s'allibera memòria un cop ja no es necessita. Amb aquest últim programa fem una anàlisi més exhaustiva, monitoritzant mètriques interessants que ens permetin entendre els *bottlenecks* d'E/S i de *swap*. Aquí afegim i descartem mètriques i eines per acabar d'enfocar el nostre estudi en un programa de IA.

3.3.3 Anàlisi d'un programa d'IA

El programa està escrit en C i es fan lectures de 2 fitxers (657.42 MiB en total). En aquesta etapa ja tenim els coneixements necessaris per a comprendre les mètriques obtingudes amb les eines. Després d'estar buscant entre diferents programes, triem una xarxa neuronal de tipus Perceptró amb tres capes de nodes: capa d'entrada, capa oculta, i capa de sortida. Aquesta utilitza el dataset MNIST. Aquest conté un conjunt d'imatges de dígits manuscrits de 0 a 9 en grids de 32x32 píxels. Com veiem a la figura 3, aquest està compost per 1s i 0s, creant la imatge d'un 0, i a la següent línia indicant quin número hi ha.

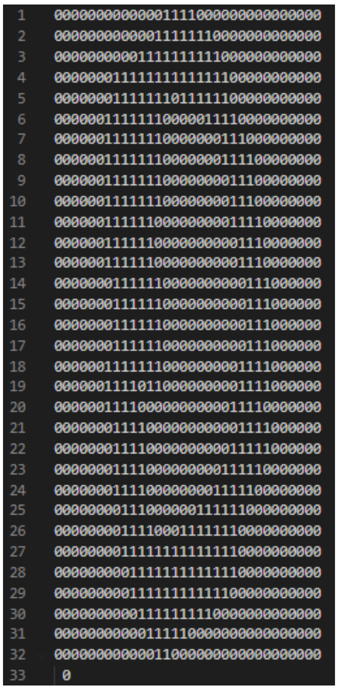

Fig. 3: Mostra del dataset d'entrenament MNIST(0)

Aquest programa l'anomenarem "MNIST2". MNIST2 és una adaptació del codi ofert a la UAB (Universitat Autònoma de Barcelona) a les pràctiques de l'assignatura de CAP (Computació d'Altes Prestacions). Les adaptacions del codi han estat les següents:

- Modificació del nombre de patrons a reconèixer del fitxer optdigits.tra. Replicant els existents de 1934 a 768.000.
- Modificació del nombre de neurones d'entrada (NUMIN) del fitxer common.h, de 1934 a 650.000.
- Modificació dels epochs realitzats a l'entrenament a l'arxiu nn-main.c, de 100.000 a 10.

S'ha triat aquest codi perquè molts programes d'IA triguen moltes hores a ser executats, i realitzant aquestes modificacions al codi aconseguim monitoritzar mètriques interessants sense que el resultat final de reconeixement de números varii gaire i executar el programa en pocs minuts. D'aquesta manera podem realitzar execucions més ràpides.

4 EINES DE MONITORITZACIÓ UTILITZADES

En aquest apartat farem una descripció de les eines utilitzades per a l'obtenció de mètriques interessants per a l'estudi de possibles *bottlenecks* en programes de IA. Totes elles són gratuites. Per a una detallada descripció del significat de cadascuna de les mètriques interessants per al nostre estudi de cada eina, mirar l'apèndix. Ve acompanyat de l'explicació aquí realitzada per a contextualitzar millor les eines amb les mètriques.

4.1 Iotop

Una eina que ens permet monitoritzar fàcilment diferents detalls d'utilització d'E/S a disc i ens mostra aquests en una taula dels diferents processos del kernel que que generen les E/S. Les dades es mostren en temps real i s'actualitzen periòdicament (cada segon per defecte). Per a utilitzar aquesta eina necessitem permisos de superusuari (root). per a què només es mostrin els processos que realment generen E/S hem d'utilitzar l'opció -o". La comanda utilitzada és la següent: sudo iotop -o.

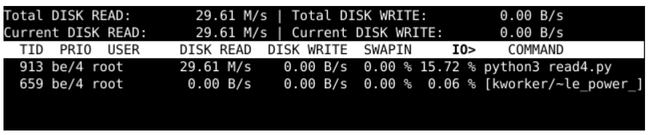

Fig. 4: Eina iotop

4.2 Top

L'eina top ens ofereix un conjunt de mètriques relacionades amb el temps d'activitat i càrrega mitja del sistema, estat de tasques, estats de la CPU, memòria física del sistema, memòria virtual i una taula a la que trobem els processos del sistema amb diferents dades com l'ús de la CPU i memòria, PID, usuari... Utilitzem l'opció -d 1 (delay) per a què l'informació s'actualitzi cada segon per a què puguem veure les dades en temps real. La comanda utilitzada és la següent: top -d 1.

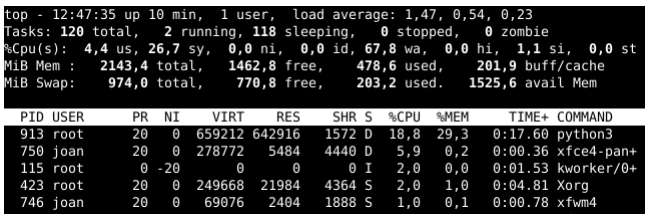

Fig. 5: Eina top

4.3 Iostat

Iostat és una eina que ens proporciona informació detallada sobre la CPU i els diferents discpositius d'emmagatzematge utilitzats al sistema. S'encarrega de monitorar la càrrega d'E/S. L'eina ve inclosa dins el paquet de sysstat, junt amb altres eines de monitorització similars. Utilitzem l'opció -x 1"per a què ens mostri estadístiques ampliades i s'actualitzi cada segon. La comanda utilitzada és la següent: iostat -x 1.

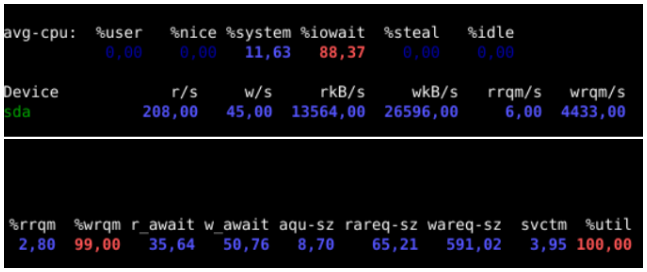

Fig. 6: Eina iostat

4.4 Vmstat

Aquesta eina ens ofereix estadístiques i dades del sistema. Ens reporta sobre processos, crides d'E/S, ús de memòria i *swap*, paginació, ús de CPU i estats del sistema. Una eina molt útil per a poder veure amb exactitut quan el sistema requereix més memòria que la física i emmagatzema dades al disc dur fins que es tornin a necessitar. De manera que la informació referent a el *swap* està relacionada amb la mètrica "SWAPIN"de l'eina iotop i les contingudes a "MiB Swap"de l'eina top. Utilitzem l'opció "1"per a què s'actualitzin les dades cada segon. La comanda utilitzada és la següent: vmstat 1.

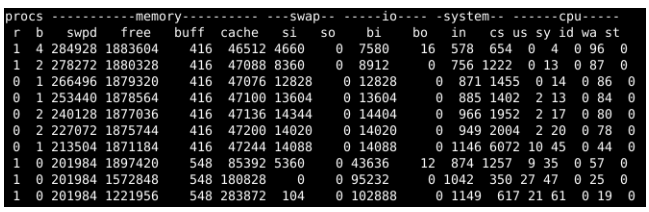

Fig. 7: Eina vmstat

4.5 Perf-stat

Aquesta eina només l'utilitzem per a calcular el temps d'execució de les diferents execucions realitzades amb diferents quantitats de memòria física administrada a màquines virtuals. Observem a la figura 8 el resultat obtingut al utilitzar perf-stat amb la següent comanda: perf stat [executable].

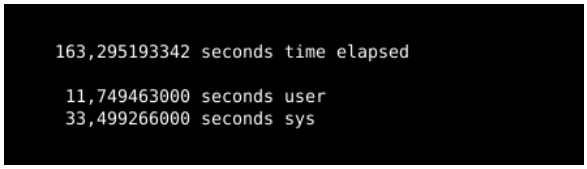

Fig. 8: Eina perf-stat

4.6 Darshan

Darshan és una eina de caracterització de E/S per HPC escalable. Per defecte està pensada per HPC i codis que utilitzin MPI, però es pot configurar per a utilitzar secuencialment sense MPI. Aquesta eina la utilitzem per a comprovar els MB en lectures a disc efectuats, l'ample de banda d'operacions d'E/S i la quantitat de temps gastat en operacions d'E/S. Aquestes 3 mètriques sense tenir en compte el *swap*. Només es podrà utilitzar en el programa d'IA, ja que aquesta eina funciona amb llenguatges C/C++, i els altres 2 estan escrits en python.

4.7 Zram i Zswap

Són dos mòduls del kernel de Linux que optimitzen el rendiment del sistema evitant la paginació a disc, així reduint el nombre d'accessos que es fa a disc. Tenen un funcionament similar, expliquem les seves diferències:
Zram crea un bloc dins de la memòria RAM per a utilitzarla com una memòria d'intercanvi de *swap*. Quan la resta de la memòria RAM estigui saturada, es comprimirà una part i es paginarà al bloc anteriorment creat.
Zswap funciona d'una manera similar, la principal diferència és que necessita un dispositiu d'emmagatzematge. La partició que crea Zswap només pren informació de la RAM que tingui una bona tassa de compressió i que no sigui susceptible a donar errors (quan la RAM té problemes d'espai). En cas contrari, s'envia a la zona d'intercanvi *swap* del dispositiu.

5 RESULTATS

Els resultats obtinguts fruit de la monitorització i observabilitat amb les eines esmentades, són extensos. A la mostra dels resultats, algunes mètriques són omeses perquè no aporten informació que necessitem o es repeteix la informació d'altres mètriques. Separem els resultats per les diferents etapes d'investigació i per eines amb les seves pròpies mètriques. En el cas de les mètriques de vmstat, utilitzem les següents icones per mostrar baix nombre de blocs de dades transmesos i per un elevat nombre (respectivament): $\downarrow - \uparrow$.

5.1 Programa de crides intensives d'E/S (read3)

Mostrem els resultats obtinguts amb el programa read3. Aquest fa una lectura de 39,9 MiB del fitxer docf.txt de 4,38 GB. En aquest cas, volem que sobri memòria per a no provocar *swap* i observar el temps que la CPU espera per operacions d'E/S. De manera que administrem 2300 MB de memòria.

TAULA 1: RESULTATS IOTOP A READ3

Iotop			
RAM	IO	SWAPIN	DISK READ
2300 MB	40%	0%	molt elevat

TAULA 2: RESULTATS TOP A READ3

Top					
RAM	id	%MEM	MiB Swap	wa	%CPU
2300 MB	0%	0,4%	0/974	40%	60%

TAULA 3: RESULTATS IOSTAT A READ3

Iostat (cpu)				
RAM	%user	%system	%idle	%iowait
2300 MB	20%	40%	0%	40%
Iostat (disk)				
RAM	%rrqm	%wrqm	await	%util
2300 MB	84%	pics 75%	2	80%

TAULA 4: RESULTATS VMSTAT A READ3

vmstat (swap + io)				
RAM	si	so	bi	bo
2300 MB	0	0	↑↑↑	↓↓↓

TAULA 5: RESULTATS PERF-STAT A READ3

perf-stat	
RAM	Temps d'execució
2300 MB	410,60 s

Aspectes a destacar dels resultats:

- Gran espera (40% del temps) de la CPU degut a operacions d'E/S a disc [IO (taula 1), wa (taula 2), %iowait (taula 3)].
- Molt elevada lectura de disc, ja que el programa només llegeix d'un fitxer [DISK READ (taula 1), bi (taula 4)].
- Molt baix ús de memòria (0,4%) [%MEM (taula 2)].
- No hi ha *swap* [SWAPIN (taula 1), si (taula 4), so (taula 4)].
- Gran percentatge de sol·licituds de lectura en espera abans de ser enviades a disc [%rrqm (taula 3)].
- Gran percentatge de temps transcorregut des de que s'envien sol·licituds d'E/S a disc fins a ser ateses [%util (taula 3)].

5.2 Programa d'E/S a causa del *swap* (read4)

El programa read4 fa una lectura de 3,9 MiB del fitxer docf.txt de 4,28 GB. En aquest cas, volem prendre mètriques en diferents execucions amb diferents quantitats de memòria física per a comprovar el seu comportament amb *swap*. Per causes de còmput, necessita 3500 MB de memòria. S'observen els resultats amb quantitats menors de memòria.

TAULA 6: RESULTATS IOTOP A READ4

Iotop			
RAM	IO	SWAPIN	DISK READ
2000 MB	20%	90%	molt elevat
2200 MB	20%	85%	molt elevat
2300 MB	20%	20%	molt elevat
2500 MB	20%	15%	molt elevat
3000 MB	20%	2%	molt elevat
3500 MB	20%	0%	molt elevat

TAULA 7: RESULTATS TOP A READ4

Top					
RAM	id	%MEM	MiB Swap	wa	%CPU
2000 MB	0%	89%	246/974	89%	15%
2200 MB	0%	89%	233/974	55%	30%
2300 MB	0%	89%	180/974	30%	60%
2500 MB	0%	80%	65/974	21%	65%
3000 MB	0%	65%	51/974	20%	75%
3500 MB	0%	52%	6,5/974	20%	75%

TAULA 8: RESULTATS IOSTAT A READ4

Iostat (cpu)				
RAM	%user	%system	%idle	%iowait
2000 MB	2%	13%	0%	85%
2200 MB	4%	26%	0%	55%
2300 MB	15%	45%	0%	30%
2500 MB	20%	45%	0%	21%
3000 MB	20%	55%	0%	20%
3500 MB	20%	55%	0%	20%
Iostat (disk)				
RAM	%rrqm	%wrqm	await	%util
2000 MB	84%	pics 80%	3,2	90%
2200 MB	pics 82%	pics 80%	2	70%
2300 MB	pics 10%	pics 80%	1,9	65%
2500 MB	0%	pics 80%	1,5	65%
3000 MB	0%	pics 80%	1,5	65%
3500 MB	0%	pics 80%	1,5	65%

TAULA 9: RESULTATS VMSTAT A READ4

vmstat (swap + io)				
RAM	si	so	bi	bo
2000 MB	↑↑	↑↑↑	↑↑↑	↑↑↑
2200 MB	↑↑	↑↑	↑↑↑	↑↑
2300 MB	↓	↓↓↓	↑↑↑	↓↓↓
2500 MB	↓	↓↓↓	↑↑↑	↓↓↓
3000 MB	↓↓↓	↓↓↓	↑↑↑	↓↓↓
3500 MB	↓↓↓	↓↓↓	↑↑↑	↓↓↓

TAULA 10: RESULTATS PERF-STAT A READ4

perf-stat	
RAM	Temps d'execució
2000 MB	270,98 s
2200 MB	114,97 s
2300 MB	52,76 s
2500 MB	48,26 s
3000 MB	44,23 s
3500 MB	43,02 s

Aspectes a destacar dels resultats:

- Gran espera (20% del temps) de la CPU degut a operacions d'E/S a disc [IO (taula 6), wa (taula 7), %iowait (taula 8)].
- Molt elevada lectura de disc, ja que el programa només llegeix d'un fitxer [DISK READ (taula 6), bi (taula 9)].
- Al llarg de les diferents execucions amb diferents quantitats de memòria, s'observa que la quantitat de *swap* que es porta a terme està relacionat amb la saturació de la memòria física. De la mateixa manera, la quantitat de memòria virtual utilitzada també augmenta amb la saturació de memòria física. [%MEM (taula 7), SWAPIN (taula 6), MiB Swap (taula 7), si so bi bo (taula 9)].
- Gran percentatge de sol·licituds de lectura en espera abans de ser enviades a disc a les execucions amb insuficient memòria física [%rrqm (taula 8), %MEM (taula 7)].
- Gran percentatge de temps transcorregut des de que s'envien sol·licituds d'E/S a disc fins a ser ateses [%util].
- Millora del temps d'execució amb més memòria administrada i menys espera d'E/S [perf-stat (taula 10), %MEM (taula 7), SWAPIN (taula 6), MiB Swap (taula 7), si so bi bo (taula 9)].
- Millora de la latència del disc a partir de 2500 MB de memòria (1,5 ms) a gaire bé la meitat que amb 2000 MB (3,2 ms) [await (taula 8)].

5.3 Programa d'IA (MNIST2)

Aquest programa no té problemes d'E/S, però si els pot tenir per *swap* com hem vist a l'anterior programa read4. De manera que també analitzem diferents execucions amb diferents quantitats de memòria física. Es fan 657,42 MiB de lectures de dos fitxers diferents, optdigits.tra i common.h. Per raons de còmput, ús de memòria per les lectures i per a que corri adecuadament el SO sense fer ús de *swap*, el programa necessita +1000 MB (2% de *swap* amb 1000 MB).

TAULA 11: RESULTATS IOTOP A MNIST2

Iotop			
RAM	IO	SWAPIN	DISK READ
800 MB	0,05%	85%	molt elevat
900 MB	0,05%	65%	molt elevat
1000 MB	0,05%	2%	baix
1500 MB	0,05%	0%	baix
2000 MB	0,05%	0%	molt baix

TAULA 12: RESULTATS TOP A MNIST2

Top					
RAM	id	%MEM	MiB Swap	wa	%CPU
800 MB	0%	73%	500/974	92%	10%
900 MB	0%	73%	490/974	80%	30%
1000 MB	0%	73%	151/974	0%	98%
1500 MB	0%	46,70%	2/974	0%	98%
2000 MB	0%	34,90%	0,3/974	0%	98%

TAULA 13: RESULTATS IOSTAT A MNIST2

Iostat (cpu)				
RAM	%user	%system	%idle	%iowait
800 MB	2%	8%	0%	90%
900 MB	5%	25%	0%	70%
1000 MB	65%	35%	0%	0%
1500 MB	65%	35%	0%	0%
2000 MB	65%	35%	0%	0%
Iostat (disk)				
RAM	%rrqm	%wrqm	await	%util
800 MB	75%	pics 90%	10	95%
900 MB	50%	pics 78%	6	90%
1000 MB	0%	pics 60%	0	4%
1500 MB	0%	pics 60%	0	0%
2000 MB	0%	pics 60%	0	0%

TAULA 14: RESULTATS VMSTAT A MNIST2

vmstat (swap + io)				
RAM	si	so	bi	bo
800 MB	↑↑↑	↑↑↑	↑↑↑	↑↑↑
900 MB	↑↑↑	↑↑↑	↑↑↑	↑↑↑
1000 MB	↓	↓	↓	↓
1500 MB	↓	↓	↓	↓
2000 MB	0	0	↓↓↓	↓↓↓

TAULA 15: RESULTATS PERF-STAT A MNIST2

perf-stat	
RAM	Temps d'execució
800 MB	9388,80 s
900 MB	2506,88 s
1000 MB	356,47 s
1500 MB	347,76 s
2000 MB	346,59 s

TAULA 16: RESULTATS DARSHAN A MNIST2

Darshan (no swap)			
RAM	MiB lectures	Vel. transferència	Temps en E/S
800 MB	657.42	11,17 MiB/s	58,80 s
900 MB	657.42	13,5 MiB/s	49,63 s
1000 MB	657.42	22,97 MiB/s	28,53 s
1500 MB	657.42	23,07 MiB/s	27,43 s
2000 MB	657.42	23,01 MiB/s	28,57 s

Aspectes a destacar dels resultats:

- Poca espera (0,05% del temps) de la CPU degut a operacions d'E/S a disc per accedir al fitxer on es troben els models d'entrenament (IO), però amb poca memòria s'eleven molt els accessos a disc a causa del *swap* [wa (taula 12), %iowait (taula 13)].
- Molt elevada lectura de disc, ja que el programa llegeix de dos fitxers, sobretot en les ocasions en les que es provoca *swap* i s'ha d'accedir a disc reiteradament [DISK READ (taula 11), bi si so (taula 14)].
- Al llarg de les diferents execucions amb diferents quantitats de memòria, s'observa que la quantitat de *swap* que es genera està relacionat amb la saturació de la memòria física. De la mateixa manera, la quantitat de memòria virtual utilitzada també augmenta amb la saturació de memòria física. [%MEM (taula 12), SWAPIN (taula 11), MiB Swap (taula 12), si so bi bo (taula 14)].
- Gran percentatge de sol·licituds de lectura en espera abans de ser enviades a disc a les execucions amb insuficient memòria física [%rrqm (taula 13), %MEM (taula 12)].
- Gran percentatge de temps transcorregut des de que s'envien sol·licituds d'E/S a disc fins a ser ateses a les execucions amb insuficient memòria física [%util (taula 13), %MEM (taula 12)].
- Millora del temps d'execució amb més memòria administrada i menys espera d'E/S [perf-stat (taula 15), %MEM (taula 12), SWAPIN (taula 11), MiB Swap (taula 12), si so bi bo (taula 14)].
- Quant més *swap* té l'execució, la velocitat de transferència disminueix i el temps en E/S augmenta [SWAPIN (taula 11), Vel. transferència Temps en E/S (taula 16)].
- S'observa un alt decrement en la latència en quant deixa de produir-se *swap* amb 1000 MB [await (taula 13), SWAPIN (taula 11)].

5.4 Optimitzacions programa d'IA amb Zswap i Zram

Prenem l'execució de 900 MB, ja que és un cas especial. El seu millor temps d'execució (370 s) dista molt de la seva pitjor execució (2500 s), sent l'únic cas en el qual ens hem trobat amb una diferència major a 20 s. El programa no canvia respecte a l'anterior experimentació.

TAULA 17: RESULTATS IOTOP A MNIST2 AMB OPTIMITZACIONS

Iotop			
Versió	IO	SWAPIN	DISK READ
900 MB	0,05%	65%	molt elevat
zswap	0,05%	10%	molt elevat
zram	0,05%	5%	molt elevat
zswap + zram	0,05%	40%	molt elevat

TAULA 18: RESULTATS TOP A MNIST2 AMB OPTIMITZACIONS

Top					
Versió	id	%MEM	MiB Swap	wa	%CPU
900 MB	0%	73%	490/974	80%	30%
zswap	0%	56%	486/974	15%	92%
zram	0%	73%	250/1074	7%	95%
zswap + zram	0%	52%	536/1074	30%	65%

TAULA 19: RESULTATS IOSTAT A MNIST2 AMB OPTIMITZACIONS

Iostat (cpu)				
Versió	%user	%system	%idle	%iowait
900 MB	5%	25%	0%	70%
zswap	43%	43%	0%	14%
zram	48%	47%	0%	5%
zswap + zram	22%	48%	0%	30%
Iostat (disk)				
Versió	%rrqm	%wrqm	await	%util
900 MB	50%	pics 78%	6	90%
zswap	20%	pics 78%	10	30%
zram	0%	pics 78%	10	10%
zswap + zram	30%	pics 78%	10	80%

TAULA 20: RESULTATS VMSTAT A MNIST2 AMB OPTIMITZACIONS

vmstat (swap + io)				
Versió	si	so	bi	bo
900 MB	↑↑↑	↑↑↑	↑↑↑	↑↑↑
zswap	↓	↓↓	↓	↓↓↓
zram	↑↑↑	↑↑↑	↓	↓↓↓
zswap + zram	↓	↓	↓	↓

TAULA 21: RESULTATS PERF-STAT A MNIST2 AMB OPTIMITZACIONS

perf-stat	
Versió	Temps d'execució
900 MB	2506,88 s
zswap	446,19 s
zram	386,46 s
zswap + zram	670,46 s

Aspectes a destacar dels resultats:

- Observem que tant l'optimització de Zram com la de Zswap milloren el rendiment, els accessos a disc i lús de la memòria i *swap*. Ajuntant les dues optimitzacions obtenim un speed up molt favorable respecte la versió original també, però inferior a les altres dues, sent més òptima la de zram (perf-stat, %MEM, SWAPIN, MiB Swap, si, so, bi, bo).
- Pel que fa a mètriques, zram provoca que top identifiqui l'espai de memòria reservat per a Zram com a memòria virtual, restant-la de la memòria física (MiB Swap).
- Augmenta considerablement la quantitat de CPU utilitzada a l'eliminar el *swap* (%CPU, SWAPIN, si, so).
- Tant Zram com Zswap provoquen aturar el *swap*, però vmstat identifica que hi ha *swap* al accedir a la secció reservada de la RAM per Zram, però comprovem que no hi ha blocs d'entrada ni sortida a disc (si, so, bi, bo, SWAP).

6 Discussió

Un cop extrets els resultats, observem diferents tendències entre les E/S, memòria física, memòria virtual i el temps d'execució:

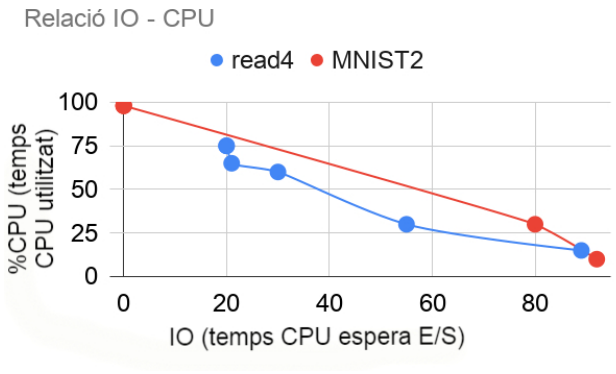

Fig. 9: Relació IO - CPU

A la figura 9 comprovem l'impacte negatiu que generen les E/S sobre el rendiment de la CPU. Com s'observa, a mida que hi ha més E/S a disc, el rendiment de la CPU disminueix dràsticament, fins al punt de no poder gaire bé seguir amb el programa esperant a carregar a memòria les dades emmagatzemades a disc. Veiem com les E/S poden provocar problemes depenent de la quantitat de memòria subministrada i de la quantitat de *swap* que es genera. En els casos en què amb MNIST2 la CPU està treballant al 100%, no s'observa cap problema d'espera per E/S. Amb read4, quan es compta amb la memòria necessària, la CPU continua esperant per E/S, ja que hi manquen operacions de còmput. Els valors de la gràfica els obtenim de les mètriques: wa i %CPU de l'eina top.

En les diferents versions de tots els programes amb els que s'ha experimentat, s'ha apreciat que el moment en el qual es comença a fer *swap* és quan la memòria física que

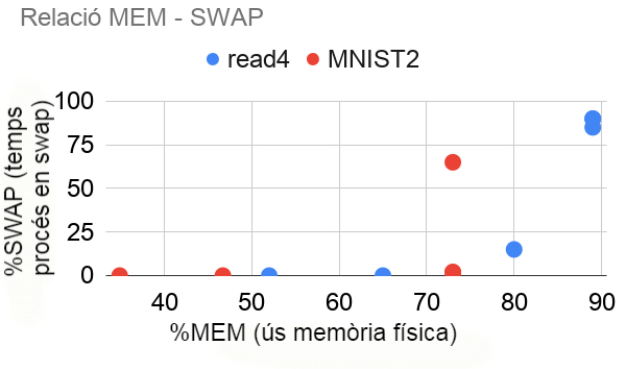

Fig. 10: Relació MEM - SWAP

administrem comença a omplir-se, però no al 100%. A la figura 10 s'observa el rang en el qual es comença a fer *swap*. És aproximadament a partir de quan s'arriba a fer ús del 70% de la RAM que es comença a fer ús de la memòria virtual. Pot ser una bona tècnica per a prevenir que s'ompli tota i hi hagi un problema de memòria que no permeti al programa seguir amb la seva execució. Per a veure aquest comportament ens fixem en les mètriques: %MEM de l'eina top i SWAPIN de l'eina iotop.

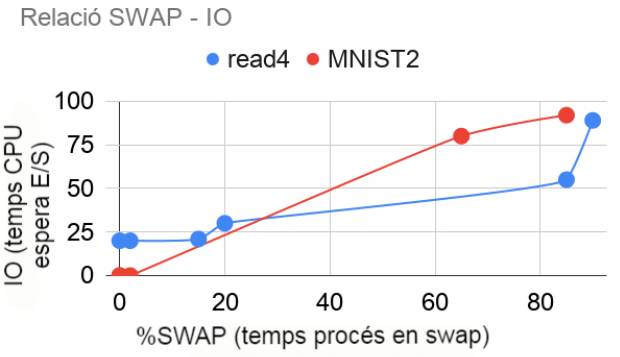

Fig. 11: Relació SWAP - IO

A la figura 11 podem observar la relació que hi ha entre les mètriques de les eines utilitzades que ens informen entre la quantitat de *swap* i el percentatge de temps que la CPU està esperant per instruccions d'E/S. En el programa read4 ja teniem a la CPU esperant sense problemes de *swap* (20%), però observem com a mida que hi ha *swap*, es crea encara més espera d'E/S. En el cas de MNIST2, tot el problema d'E/S ve generat pel *swap*. Són dades en les quals és molt important fixar-se, ja que encara que no tinguem problemes d'operacions d'E/S (sense comptar amb el *swap*) en un programa, la memòria administrada és un factor molt important en el rendiment final en els casos observats. En aquesta gràfica ens fixem en les mètriques: wa de l'eina top i SWAPIN de l'eina iotop.

Per acabar, separem en dues gràfiques com afecta finalment la quantitat de temps que la CPU espera per instruccions d'E/S al temps d'execució total del programa. S'observa a les figures 12 i 13 com es multiplica el temps d'execució. Per observar aquest comportament ens fixem en les mètriques: wa de l'eina top i el temps d'execució de l'eina perf-stat.

Als resultats es veu com, mètriques de diferents eines que

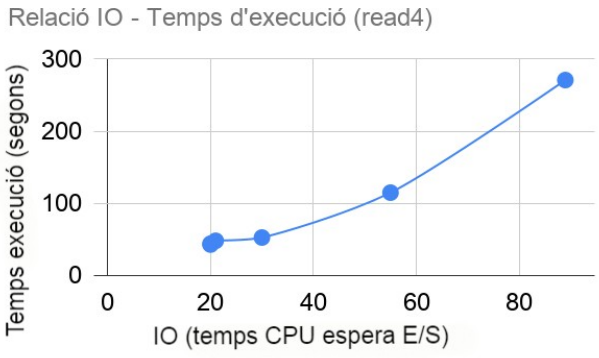

Fig. 12: RELACIÓ IO - TEMPS D'EXECUCIÓ (READ4)

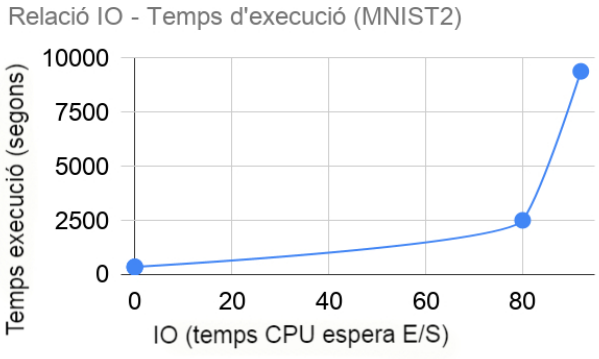

Fig. 13: RELACIÓ IO - TEMPS D'EXECUCIÓ (MNIST2)

haurien de donar els mateixos resultats, mostren valors diferents. La raó de que això ocorri l'atribuïm a que les eines estan prenent les mètriques de diferents capes i en instants diferents. De manera que és important executar múltiples vegades el programa que volem analitzar amb eines. Diferents execucions ens poden donar diferents valors, a més, depenent del moment de l'execució també varien aquests valors. Un exemple el podem trobar amb MNIST2 a l'execució amb 900MB de memòria. El seu millor temps d'execució (370 s) dista molt de la seva pitjor execució (2500 s), tot i que aquest és l'únic cas en el qual ens hem trobat amb una diferència major a 20 s. Zram i Zswap empitjoren pocs segons el millor temps del cas esmentat (386 i 446 respectivament), però es mantenen estables al llarg de les execucions.

7 CONCLUSIONS

En aquest treball hem fet un estudi amb el fi de donar una guia per a què programadors puguin conèixer diferents eines de monitorització de recursos i les mètriques que s'extreuen amb elles. Com utilitzar aquestes eines i com interpretar-les, amb diferents exemples per a poder entendre com poden variar les dades obtingudes en diferents casos. En concret pot ser un informe útil per a aquells que vulguin utilitzar programes d'IA. Aquests tipus de programes han d'accedir a molts fitxers per la seva naturalesa, i això comporta moltes crides d'E/S a dispositius d'emmagatzematge. Després d'aquest estudi veiem com aquestes poden afectar el rendiment del programa i la importància que té saber identificar un *bottleneck* d'aquest tipus. També hem vist la importància que té la quantitat de memòria física que se li administra a la màquina, ja que aquesta pot provocar *swap* i així accentuar el problema de *bottleneck* amb les E/S.

REFERÈNCIES

[1] Rouhiainen, L. (2018). Inteligencia artificial. Madrid: Alienta Editorial.

[2] Centeno Franco, A. (2019). Deep learning. (Trabajo Fin de Grado Inédito). Universidad de Sevilla, Sevilla.

[3] Chien, S. W., Podobas, A., Peng, I. B., Markidis, S. (2020, September). tf-Darshan: Understanding Fine-grained I/O Performance in Machine Learning Workloads. In 2020 IEEE International Conference on Cluster Computing (CLUSTER) (pp. 359-370). IEEE.

[4] González Férez, M. P. (2012). Simulación concurrente y elección dinámica de estrategias para la mejora de la entrada/salida de disco (Doctoral dissertation, Universidad de Murcia).

[5] Pérez Hernández, M. D. L. S. (2003). Arquitectura multiagente para E/S de alto rendimiento en clusters (Doctoral dissertation, Informatica).

[6] González Becerril, K. (2019). Estudio Comparativo para Demostrar las Ventajas y Desventajas de las Unidades de Almacenamiento: Disco Duro y Unidad de Estado Sólido.

[7] de Usera, J. D. (2021). ¿En qué se diferencian los SSD de los HDD / discos duros?. Recuperado 9 de junio de 2021, de Hardzone website: https://hardzone.es/tutoriales/componentes/discos-duros-ssd-diferencias/

[8] Patzi Velarde, M. Modelo de optimización para mejorar el rendimiento de memoria virtual en servidores GNU/Linux (Doctoral dissertation).

[9] Bottallo, D. (2011). ADMINISTRACIÓN DE LA MEMORIA. Recuperado 9 de junio de 2021, de Facultad de Ciencias Exactas, Ingeniería y Agrimensura website: https://www.fceia.unr.edu.ar/ diegob/so/presenta/09-Memoria.pdf

[10] Memory paging. (2021). Recuperado 9 de junio de 2021, de Wikipedia website: https://en.wikipedia.org/wiki/Memory_paging

[11] Virtual Memory in Operating System. (2011). Recuperado 9 de junio de 2021, de Meherchilakalapudi website: https://meherchilakalapudi.wordpress.com/2011/12/18/virtual-memory-in-operating-system/

[12] Lamas Daviña, A. (2011). Evaluación de prestaciones de aplicaciones paralelas en diferentes configuraciones de memoria en arquitectura NUMA (Doctoral dissertation, Universitat Politècnica de València).

[13] Perera, I. (2018). Linux Performance Observability Tools. Recuperado 5 de marzo de 2021, de medium website: https://medium.com/@chrishantha/linux-performance-observability-tools-19ae2328f87f.

APÈNDIX

A EXPLICACIÓ MÈTRIQUES DE LES EINES UTILITZADES

A.1 Iotop

Una eina que ens permet monitoritzar fàcilment diferents detalls d'utilització d'E/S a disc i ens mostra aquests en una taula dels diferents processos del kernel que que generen les E/S. Les dades es mostren en temps real i s'actualitzen periòdicament (cada segon per defecte). Per a utilitzar aquesta eina necessitem permisos de superusuari (root). per a què només es mostrin els processos que realment generen E/S hem d'utilitzar l'opció -o". La comanda utilitzada és la següent: sudo iotop -o.

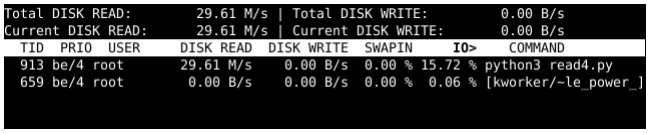

Fig. 14: EINA IOTOP

Les dades que observem d'aquesta eina són les següents, les podem observar a la figura 14:

- IO: Ús de cada procés d'E/S. Exactament mostra la fracció de cada procés gastat en E/S. Si d'entrada veiem que aquest valor és elevat, podem tenir un *bottleneck* a les crides a disc.
- SWAPIN: Ús de *swap* de cada procés. Mostra la fracció de cada procés gastat en *swap*. Això també pot generar *bottleneck* a E/S, tot i que no sempre sigui identificat a la mètrica IO, però si amb altres mètriques similars obtingudes amb altres eines posteriorment comentades.
- DISK READ: Quantitat de dades per segon de lectura de disc.

A.2 Top

L'eina top ens ofereix un conjunt de mètriques relacionades amb el temps d'activitat i càrrega mitja del sistema, estat de tasques, estats de la CPU, memòria física del sistema, memòria virtual i una taula a la que trobem els processos del sistema amb diferents dades com l'ús de la CPU i memòria, PID, usuari... Utilitzem l'opció -d 1 (delay) per a què l'informació s'actualitzi cada segon per a què puguem veure les dades en temps real. La comanda utilitzada és la següent: top -d 1.

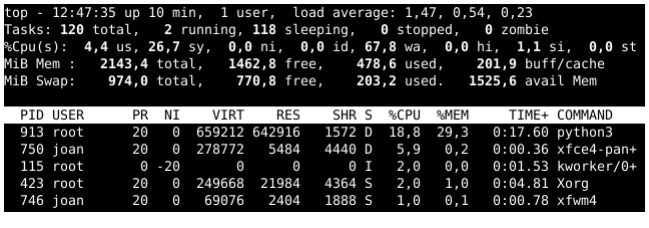

Fig. 15: EINA TOP

Les dades que observem d'aquesta eina són les següents, les podem observar a la figura 15:

- wa: Percentatge de temps de la CPU esperant per operacions d'E/S. Aquesta mètrica, també trobada amb l'eina vmstat posteriorment comentada, té en compte el temps d'espera generat per *swap*.
- %MEM: Percentatge de memòria física utilitzada per cada procés des de l'última actualització.
- MiB Swap: Ens ofereix diferents dades relacionades amb la memòria virtual disponible, lliure i utilitzada. Aquestes dades es veuen estretament relacionades amb la capacitat de fer *swap*.
- %CPU: Percentatge de CPU utilitzat per cada procés des de l'última actualització.
- id: Percentatge de CPU utilitzat en processos inactius (CPU en desús).

A.3 Iostat

Iostat és una eina que ens proporciona informació detallada sobre la CPU i els diferents discpositius d'emmagatzematge utilitzats al sistema. S'encarrega de monitorar la càrrega d'E/S. L'eina ve inclosa dins el paquet de sysstat, junt amb altres eines de monitorització similars. Utilitzem l'opció -x 1"per a què ens mostri estadístiques ampliades i s'actualitzi cada segon. La comanda utilitzada és la següent: iostat -x 1.

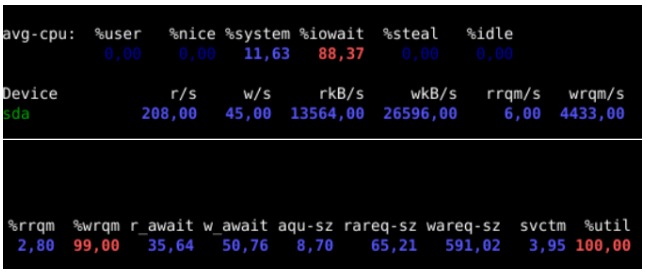

Fig. 16: EINA IOSTAT

Les dades que observem d'aquesta eina són les següents, les podem observar a la figura 16:

- %user: Percentatge de temps de CPU gasta en diferents processos (user end process).
- %sys: Percentatge de temps de CPU gasta en tasques del sistema operatiu (user end process).
- %idle: Percentatge de CPU utilitzat en processos inactius (CPU en desús).
- %iowait: Percentatge de temps de la CPU esperant per operacions d'E/S. Els resultats obtinguts coincideixen en la major part dels casos amb la mètrica "wa"de les eines top i vmstat.
- %rrqm: Percentatge de sol·licituds de lectura en espera abans de ser enviades a disc.
- %wrqm: Percentatge de sol·licituds d'escriptura en espera abans de ser enviades a disc.

- await: Mitjana del temps en milisegons d'operacions d'E/S a la cua del disc dur esperant per a ser ateses + temps dedicat a atendre-les pel disc (latència).
- %util: Percentatge de temps transcorregut durant el qual s'han emès sol·licituds d'E/S al dispositiu. Percentatge de temps duarant el qual, el dispositiu està fent almenys una operació. El disc es veu saturat si aquest valor és proper al 100%.

A.4 Vmstat

Aquesta eina ens ofereix estadístiques i dades del sistema. Ens reporta sobre processos, crides d'E/S, ús de memòria i *swap*, paginació, ús de CPU i estats del sistema. Una eina molt útil per a poder veure amb exactitut quan el sistema requereix més memòria que la física i emmagatzema dades al disc dur fins que es tornin a necessitar. De manera que la informació referent a el *swap* està relacionada amb la mètrica "SWAPIN"de l'eina iotop i les contingudes a "MiB Swap"de l'eina top. Utilitzem l'opció "1"per a què s'actualitzin les dades cada segon. La comanda utilitzada és la següent: vmstat 1.

```
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  4 284928 1883604    416  46512 4660    0  7580    16  578  654  0  4  0 96  0
 1  2 278272 1880328    416  47088 8360    0  8912     0  756 1222  0 13  0 87  0
 0  1 266496 1879320    416  47076 12828     0 12828     0  871 1455  0 14  0 86  0
 0  1 253440 1878564    416  47100 13604     0 13604     0  885 1402  2 13  0 84  0
 0  2 240128 1877036    416  47136 14344     0 14404     0  966 1952  2 17  0 80  0
 0  2 227072 1875744    416  47200 14020     0 14020     0  949 2004  2 20  0 78  0
 0  1 213504 1871184    416  47244 14088     0 14088     0 1146 6072 10 45  0 44  0
 1  0 201984 1897420    548  85392 5360    0 43636    12  874 1257  9 35  0 57  0
 1  0 201984 1572848    548 180828    0    0 95232     0 1042  350 27 47  0 25  0
 1  0 201984 1221956    548 283872  104    0 102888     0 1149  617 21 61  0 19  0
```

Fig. 17: EINA VMSTAT

Les dades que observem d'aquesta eina són les següents, les podem observar a la figura 17:

- swpd: Tamany de memòria virtual utilitzada. Des del moment en que és major a 0, significa que la memòria física és insuficient i s'està portant a terme *swap*.
- bi: Nombre de blocs rebuts per segon per dispositius de blocs (disc).
- bo: Nombre de blocs enviats per segon a dispositius de blocs (disc).
- si: Nombre de blocs rebuts per segon per dispositius de blocs deguts al *swap* (disc).
- so: Nombre de blocs enviats per segon a dispositius de blocs deguts al *swap* (disc).
- id: Percentatge de CPU utilitzat en processos inactius (CPU en desús).
- wa: Percentatge de temps de la CPU esperant per operacions d'E/S.

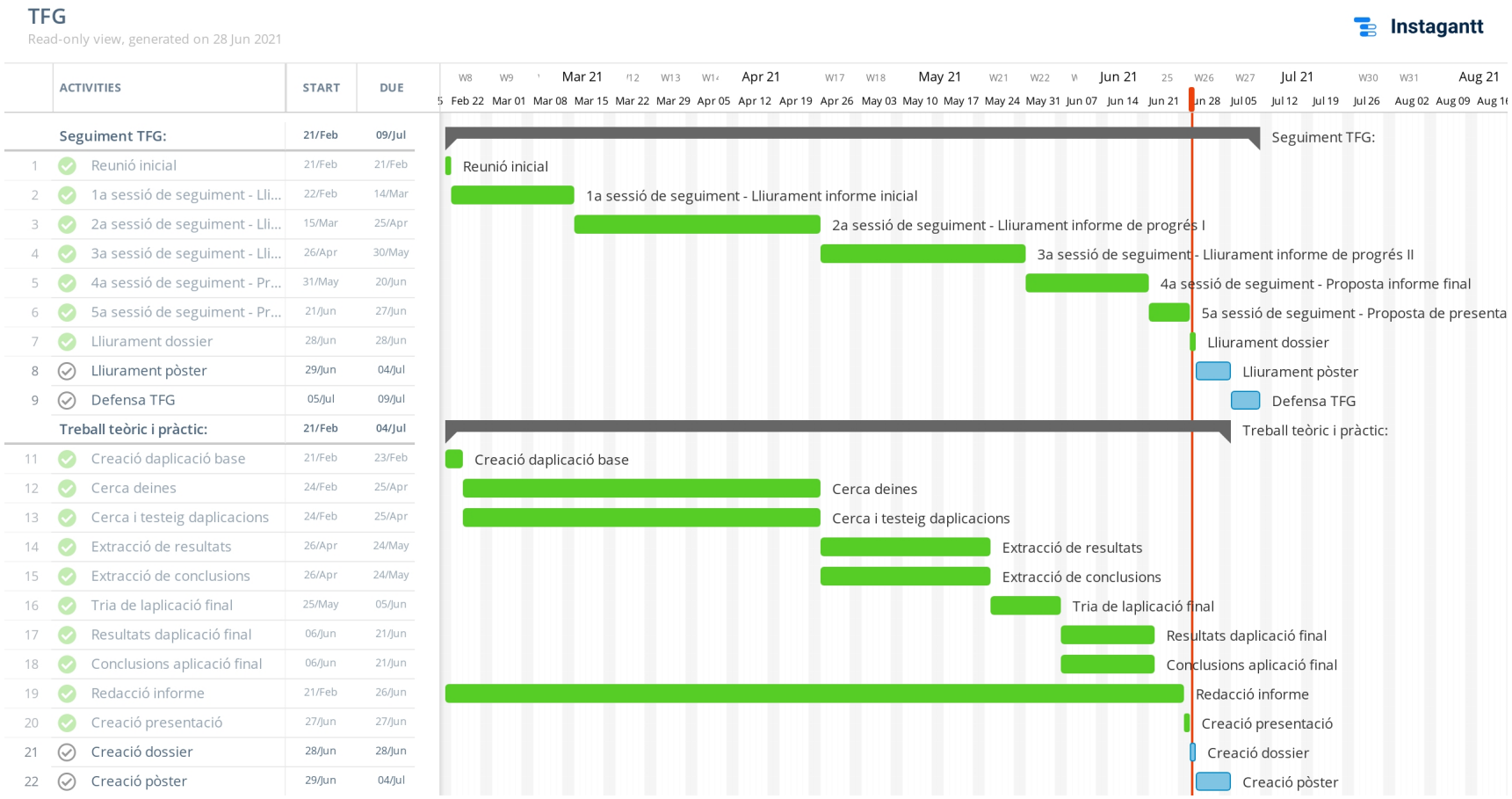

Fig. 18: DIAGRAMA D'UNA PIPELINE D'ENTRENAMENT.