Paral·lelització de workflows d'anàlisi de dades d'astronomia de raigs gamma

Yasmin L'Harrak El Awami

Resum— Aquest projecte es centra en la paral·lelització dels workflows d'anàlisi del raig gamma per a les corbes de llum i l'espectre de distribució d'energia (SED). Especialment, s'explora la detecció dels raigs gamma emesos per fenòmens astronòmics d'alta energia com forats negres i supernoves, utilitzant el telescopi MAGIC. La creixent quantitat de dades generades en aquest àmbit planteja una problemàtica significativa, exigint estratègies avançades per processar-les i analitzar-les eficaçment. Aquesta iniciativa s'adreça a millorar l'eficiència de les tasques d'anàlisi, implementant paral·lelització amb Dask, RAY i Multiprocessing per accelerar el processament dels fluxos de dades complexos i voluminosos, contribuint a l'avanç de l'astronomia de raigs gamma i la seva comprensió més profunda dels fenòmens astrofísics d'alta energia.

Paraules clau—Gammapy, Dask, Magic, corba de llum, espectre de distribució d'energia (SED), astrofísica, Cherenkov, telescopi, HTCondor, paral·lelització, Apache Spark, Python, Astropy, Centre d'investigació científica (PIC), DL3, Fits, HIVE.

Abstract— This project focuses on parallelizing workflows for gamma-ray analysis of light curves and energy distribution spectra (SED). Specifically, it explores the detection of gamma rays emitted by high-energy astronomical phenomena such as black holes and supernovae, using the MAGIC telescope. The increasing volume of data generated in this field poses a significant challenge, demanding advanced strategies for efficient processing and analysis. This initiative aims to enhance the efficiency of analysis tasks by implementing parallelization using Dask, RAY, and Multiprocessing to accelerate the processing of complex and voluminous data streams, thereby advancing gamma-ray astronomy and deepening our understanding of high-energy astrophysical phenomena.

Index Terms— Gammapy, Dask, Magic, light curve, spectral energy distribution (SED), astrophysics, Cherenkov, telescope, HTCondor, parallelization, Apache Spark, Python, Astropy, Scientific Research Center (PIC), DL3, Fits, HIVE.

1 Introducció - Context del treball

L'ASTROFÍSICA[1] és la ciència observacional, en la que els principis de la física s'apliquen a l'astronomia per estudiar els objectes i fenòmens astronòmics[2], en altres paraules, es centra en l'estudi de la formació i l'evolució de l'univers.

Les ones electromagnètiques[3] són la forma en què l'energia de la radiació electromagnètica es mou en l'espai i en el temps. Aquestes ones es representen en l'espectre electromagnètic, a més organitza les diferents ones segons la seva freqüència i, per tant, la seva energia. Sent així, les ones de ràdio les menys energètiques, i els raigs gamma[4], les ones més energètiques.

S'explora específicament l'àmbit de l'astronomia de raigs gamma, centrant-se en l'estudi de les interaccions entre les partícules generades pels fenòmens astronòmics d'alta energia que s'emeten des de forats negres, nuclis de galàxies o les explosions de supernoves. Quan els fotons i les partícules associades impacten en l'atmosfera terrestre, generen una sèrie d'esdeveniments coneguts com a radiació Cherenkov[5]. La llum de Cherenkov no és visible a ull humà, però amb l'ús de telescopis especialitzats, equipats amb càmeres i electrònica adequada, és possible detectar-la i capturar-la.

- *E-mail de contacte: 1603496@uab.cat*
- *Treball tutoritzat per: Eduardo Cesar Galobardes (Àrea d'Arquitectura i de Tecnologia de Computadors)*
- *Curs 2023/24*

Actualment, existeixen dispositius de detecció de raigs gamma integrats en satèl·lits com ara el FERMI-LAT de la NASA, o per exemple telescopis terrestres, com ara, MAGIC[6], HESS, VERITA i el FACT.

La comunitat científica que treballa amb els principals detectors terrestres de radiació Cherenkov estan impulsant el desenvolupament d'una nova generació d'instruments coneguts com a Cherenkov Telescope Array (CTA)[7]. Aquest nou conjunt de 66 telescopis, distribuïts en dos observatoris, té com a objectiu ampliar el rang d'energies observables en la detecció de raigs gamma, i analitzar-les conjuntament amb eines com Gammapy[8], un paquet de Python que permet explotar les dades de la nova generació i del nou format de dades, DL3[9], impulsats per la comunitat.

A més, es pretén augmentar el volum de dades recopilades per tots els telescopis per poder abordar qüestions científiques que encara es troben sense resposta

Per analitzar els raigs gamma, es fan servir tècniques comunes entre els científics de la comunitat. Per a l'exploració ràpida de dades i l'anàlisi profund, s'implementa el càlcul de la corba de llum[10] i l'espectre de distribució d'energia (SED)[11]. Aquests anàlisis, basats en Gammapy, es realitzen seqüencialment i es limiten a conjunts de dades d'unes poques GB.

Donat això, la motivació d'aquest projecte rau en el problema de l'augment constant de terabytes de dades, que fa que l'anàlisi dels raigs gamma es torni cada vegada més lent. Els

mètodes d'anàlisi seqüencial de corbes de llum i espectres de distribució d'energia són limitats quan es tracta de terabytes de dades recopilades per diferents telescopis. Per tant, és necessari explorar alternatives per millorar la velocitat i l'eficiència d'aquests anàlisis. La implementació d'una versió paral·lelitzada dels workflows d'anàlisi és una opció prometedora. Aquesta estratègia pot minimitzar dependències externes i aportar escalabilitat.
Donat el flux seqüencial dels workflows d'anàlisi dels raigs gamma, s'ha observat que tots segueixen un mateix patró, tal com es mostra a la Figura 1.

Figura 1: Etapes workflows

Per tant, s'ha dividit el workflow en cinc etapes per analitzar exhaustivament cadascuna i identificar els colls d'ampolla. S'ha observat que, en ambdós workflows, els dos últims blocs són els que consumeixen més temps.

Per la Corba de Llum:

- Process Observations: 65,73% del temps total
- Fitting: 32,52% del temps total

Per SED:

- Process Observations: 26,96% del temps total
- Fitting: 72,97% del temps total

S'han detectat redundàncies al bloc de Process Observations del workflow de SED. Després de refactoritzar el codi s'ha aconseguit una acceleració de 1,78x.
Després d'optimitzar els blocs més lents, s'explora la seva paral·lelització per reduir els temps d'execució. Per paral·lelitzar eficaçment, és crucial utilitzar un entorn com el Port d'Informació Científica (PIC)[12], que ofereix plataformes de còmput distribuït per optimitzar els temps d'execució mitjançant tècniques de paral·lelisme.
S'ha implementat una versió des de zero paral·lelitzada del bloc Process Observations amb Dask[13] per gestionar la paral·lelització distribuïda en el clúster d'HTCondor[14]. Pel bloc Fitting, s'activa la paral·lelització de les llibreries de Python RAY[15] i Multiprocessing[16] desenvolupades pels creadors de Gammapy, encara en fase de prototip. Es prova cada opció per determinar el millor rendiment, augmentant el nombre de CPUs disponibles a ambdós paral·lelitzacions.
Aquests han estat els resultats finals més prometedors:

Per la Corba de llum:

- Process Observations: acceleració de 7,40x amb 16 CPUs.
- Fitting (Multiprocessing): acceleració de 1,66x amb 2 CPUs.

Per SED:

- Process Observations: acceleració de 19,79x amb 16 CPUs.
- Fitting (Multiprocessing): acceleració de 2,18x amb 2 CPUs.

Els resultats de les paral·lelitzacions han estat molt prometedors, amb una acceleració significativa respecte al workflow en sèrie. Multiprocessing s'ha mostrat més eficient que RAY, ja que RAY introdueix complexitats addicionals per a configuracions distribuïdes. A més, no s'ha utilitzat per a la Corba de Llum, perquè encara és un prototip i no està completament operatiu per a tots els mètodes i funcions d'anàlisi.
Per la Corba de Llum, amb una acceleració de 7,40x utilitzant 16 CPUs exemplifica l'eficiència d'execució, mantenint una relació excel·lent entre eficiència i temps d'execució. El Fitting amb Multiprocessing també mostra una millora substancial amb una acceleració de 1,66x amb 2 CPUs.
Pel SED, s'aconsegueix una acceleració de 19,79x utilitzant 16 CPUs. El Fitting amb Multiprocessing també experimenta una acceleració notable de 2,18x amb 2 CPUs, il·lustrant una eficiència equilibrada en l'acceleració del processament.
Així doncs, es presenta una acceleració global de 2,52x pel workflow de SED i una acceleració global de 3,32x pel workflow de la corba de llum.

Finalment, es presenta l'estructura d'aquest treball. La Secció 2 per al context i els conceptes clau, la Secció 3 que aborda la problemàtica i el sistema actual, la Secció 4 amb les optimitzacions implementades, la Secció 5 amb els resultats obtinguts, i la Secció 6 que conclou i presenta les línies futures.

2 CONTEXTUALITZACIÓ

A continuació, es presenta la secció que explica els conceptes necessaris per comprendre el projecte.

2.1 DADES DL3

El format que s'utilitza per emmagatzemar les dades en aquest projecte d'anàlisi científic, és el del tercer nivell de dades CTA, anomenat DL3 i contingut en un format comú d'astronomia anomenat FITS[17], ja que es volen estandarditzar les dades per tal de poder analitzar-les conjuntament amb les d'altres instruments.
Aquest format permet gestionar grans volums de dades de manera eficient, gràcies a l'estructura reduïda, autocontinguda i amb tots els components necessaris per poder analitzar les dades.

- Taules d'events detectats per una única observació
- La capçalera que conté les metadades sobre l'observació
- Les Instruments Response Functions (IRF)[18] que determinen la calibració i paràmetres d'observació

Aquests components es troben dins de l'estructura dels fitxers FITS, és a dir, cada component es troba en un Header Data Unit (HDU)[19] diferent. A la vegada, cada HDU conté un header amb les metadades que descriuen les dades d'aquell bloc i les respectives dades científiques reals.
Pel desenvolupament d'aquest treball, es considerarà un conjunt de mostra de dades de 2000 observacions del telescopi MAGIC correctament convertides, al format DL3.

2.2 PIC

El PIC és un centre orientat a maximitzar el còmput amb paràmetres de throughput en comptes de la visió de computació d'alt rendiment. Aquest enfocament és adient per càrregues de còmput científiques conformades per tasques independents.

Addicionalment, opera en un entorn compartit i heterogeni amb diversitat tecnològica, utilitzant equips que incorporen diverses arquitectures de hardware i sistemes operatius adaptats específicament per a cada àmbit d'investigació. La infraestructura de xarxa del PIC no només facilita la comunicació entre les

màquines i suporta la distribució de treballs i la transferència de dades massives, sinó que també gestiona nivells de xarxa diferents, ajustant-se a les necessitats particulars de cada aplicació i a les limitacions de connectivitat a cada sala de càlcul.
S'analitza el sistema de còmput del PIC mitjançant un esquema per capes, que mostra la interacció entre els diferents nivells, com es mostra a la Figura 2.

Figura 2: Estructura PIC

A la base de l'organització es troba la capa física, amb màquines que inclouen diverses arquitectures i configuracions. Aquestes màquines estan repartides entre dues sales de còmput amb una xarxa que les connecta, i amb la particularitat que unes estan en racks verticals amb refrigeració per aire, i d'altres estan en racks horitzontals refrigerades mitjançant oli mineral.
Per sobre d'aquesta capa, s'instal·la la capa de HTCondor[25] per gestionar la distribució i l'execució eficient dels treballs distribuïts. La capa de HTCondor també inclou polítiques de compartició dels recursos a nivell d'usuari, de grups o experiments, que en condicionen la disponibilitat i en monitoren l'ús.
Més recentment el PIC ha incorporat una capa de més alt nivell mitjançant el paquet Dask[26] per facilitar la paral·lelització distribuïda i l'escalabilitat dinàmica de les tasques computacionals, ajustant els recursos segons les necessitats específiques de la investigació. La comunicació entre aquestes dues capes es realitza mitjançant la interfície HTCondorCluster[27], que connecta de manera eficaç les funcionalitats de Dask i HTCondor, integrant-les per un ús coherent i eficient dels recursos del PIC en l'execució de treballs distribuïts, permetent un monitoratge continu.
A més, Dask forma part de l'immens ecosistema de Python i també ofereix un sistema d'aprenentatge automàtic que millora i suporta llibreries com NumPy[29] i Pandas[30], àmpliament utilitzades en la comunitat científica per a la modelització de dades gràcies a la seva integració nativa amb Python.

2.2.1 DASK

Dask permet gestionar grans volums de dades distribuint la càrrega de treball en múltiples màquines del clúster HTCondor, cosa que és especialment útil per disminuir la gran càrrega de còmput de manera eficient. Amb Dask, es pot convertir bucles seqüencials en tasques paral·leles, maximitzant l'eficiència del processament de dades.
S'estructura en tres components principals: el client, el scheduler i els workers. El desenvolupador es comunica amb el client de Dask per enviar les tasques al clúster. El scheduler, que és el cervell del clúster, rep aquestes tasques i les distribueix de manera eficient entre els workers.
La distribució es realitza de la següent manera:

- Primerament, es distribueixen N tasques entre els N workers configurats.
- A mesura que un worker finalitza la seva tasca, el scheduler li assigna una de nova.

Així, tot i que globalment un worker pot haver tingut més tasques que un altre, el procés es porta a terme de manera eficient, ja que s'assegura el balanceig de la càrrega.

Els workers són les màquines que executen les tasques assignades pel scheduler. Un cop completades, els resultats dels workers es retornen al client, tal com es mostra a la Figura 3.
La connexió establerta facilita la gestió de tasques de gran escala de manera eficient i distribuïda, aprofitant els recursos del clúster configurat.

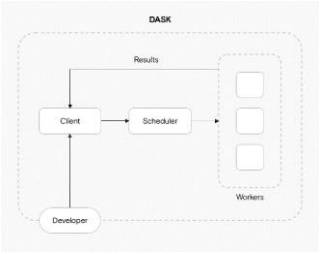

Figura 3: Esquema Dask

2.2.2 Configuració HTCondor

La configuració del clúster HTCondor ha estat la següent:

- Crear un clúster de Dask per tal de definir un conjunt de recursos sobre el qual executarem els nostres treballs. Aquest clúster, segons la implementació i disseny dels sistemes del PIC, correrà sobre d'una infraestructura orquestrada per HTCondor. La definició del clúster es fa mitjançant una classe que fa d'interfície, la classe *HTCondorCluster*. Per definir el tipus de màquina bàsica del nostre clúster Dask, o també anomenat worker, es defineixen com a paràmetres d'entrada el nombre de cores, és a dir, el nombre de nuclis de CPU, per node/màquina del clúster, juntament amb la memòria RAM i l'espai en disc per cada nucli. A més, s'assegura que les variables d'entorn de l'entorn actual es transfereixin als nodes de treball.
- A continuació, al perfil de màquina del clúster que s'ha definit, s'aplica el mètode scale per definir l'escala del clúster. Aquest mètode rep un paràmetre d'entrada que és el nombre de rèpliques d'aquesta màquina bàsica definida, amb les mateixes característiques especificades. Escalar el clúster implica que es sol·licita a HTCondor que gestioni i proporcioni els recursos necessaris per assolir el nombre de nodes desitjat.
- Finalment, es crea un client de Dask que es connecta al clúster configurat. Aquest client permet la interacció amb el clúster, és a dir, permet enviar i gestionar treballs de processament distribuïts a través del clúster HTCondor, així com accedir a la monitorització de l'activitat del clúster amb un panell de mètriques predefinides.

3 ANÀLISI SISTEMA ACTUAL

Per tal d'exemplificar uns tipus d'anàlisi molt comuns entre els científics de la comunitat de raigs gamma, tant per la ràpida exploració de dades com per un anàlisi més profund, s'ha triat la implementació del càlcul de la corba de llum[21] i l'espectre de distribució d'energia (SED)[22].
La raó per determinar aquest tipus d'anàlisi ha estat perquè la corba de llum revela els patrons de variabilitat temporal en l'emissió dels raigs gamma, facilitant l'estudi d'esdeveniments transitoris i dinàmics en l'univers i el SED permet conèixer més sobre la naturalesa de la font, la seva composició, temperatura, entre altres característiques.

Aquests anàlisis basats en Gammapy es fan seqüencialment, i actualment es limiten a un conjunt de dades de l'ordre d'unes poques GB. Ara bé, davant el repte que presenta CTA amb un volum de dades DL3 generades anualment de l'ordre de TB, es replanteja l'estructura de funcionament que hi havia.
En aquest projecte, que utilitza 2000 observacions d'aproximadament 2 GB cadascuna, filtrades per la regió del cel d'interès per a l'anàlisi, es redueixen a 155 observacions. El temps necessari per al workflow en sèrie de la corba de llum és de 190,69 segons, i per al workflow en sèrie d'SED és de 1262,79 segons.
Donat el flux seqüencial dels workflows d'anàlisi dels raigs gamma, s'ha observat que tots segueixen un mateix patró, tal com es mostra a la figura 1.

- **Data Selection**: realitza la selecció de les dades, és a dir, aquelles que es troben a la regió sotmesa a anàlisi.
- **Analysis Config**: s'encarrega d'efectuar la configuració de l'anàlisi.
- **Get Observations**: s'encarrega de recuperar les observacions presents a la regió de les dades MAGIC.
- **Process Observations**: responsable de processar les observacions i dur a terme l'anàlisi.
- **Fitting**: realitza el procés de l'ajust dels models a les dades observades.

Una possible estratègia a explorar és la paral·lelització del flux de treball per fer l'anàlisi. Això implica dividir els processos i que s'executin de manera simultània en diferents recursos computacionals. Permet dur a terme l'anàlisi de manera més eficient, aprofitant tots els recursos disponibles i reduint el temps.
Per poder aplicar una possible estratègia de paral·lelització i poder dividir en processos, s'ha vist necessari agrupar el codi per etapes, ja que és una bona pràctica per l'organització clara, facilita el manteniment i la identificació d'errors. A més, ajuda a analitzar exhaustivament i identificar quines parts del procés hi ha presència del coll d'ampolla i es consumeix més temps.
Per a cada etapa, es compta amb una mètrica de temps. Per a cada workflow, es registra i es suma el temps de les etapes. En tots els casos, la mesura es realitza en segons.
S'ha observat que, en ambdós workflows, els blocs que requereixen més temps són els dos últims, tal com es mostra a la Figura 4.

Figura 4: Etapes problemàtiques

TEMPS (s) CORBA LLUM	
	SÈRIE
DATA SELECTION	2,73
ANALYSIS CONFIG	0,01
GET OBSERVATIONS	0,6
PROCESS OBSERVATIONS	**125,34**
FITTING	**62,01**
TOTAL CONJUNT TASQUES	190,69

Taula 1: Corba llum sèrie

TEMPS (s) SED	
	SÈRIE
DATA SELECTION	0,15
ANALYSIS CONFIG	0,12
GET OBSERVATIONS	0,49
PROCESS OBSERVATIONS	340,48
FITTING	**921,55**
TOTAL CONJUNT TASQUES	1262,79

Taula 2: SED sèrie

Tal com es mostra a les Taules 1 i 2, els blocs que més temps consumeixen en tots dos workflows són el Process Observations i el Fitting.

Per la Corba de Llum:

- **Process Observations**: 125,34 segons, representant el 65,73% del temps total.
- **Fitting**: 62,01 segons, representant el 32,52% del temps total.

Per SED:

- **Process Observations**: 340,48 segons, un 26,96% del temps total
- **Fitting**: 921,55 segons, representant el 72,97% del temps total.

Process Observations: conté un bucle que s'executa per cada observació astronòmica. En aquest bloc es processen i modifiquen els datasets corresponents a cada observació, i s'afegeixen a una col·lecció final. Les operacions que s'hi realitzen inclouen:

- *Creació de datasets*. Cada observació requereix la creació d'un dataset inicial.
- *Integració de dades de Fons*. Es processen les dades de fons i s'integren als datasets corresponents.
- *Aplicació de màscares de seguretat*. Es duu a terme l'aplicació de màscares per assegurar la qualitat i fiabilitat de les dades considerades.

Aquest procés iteratiu i intensiu per cada observació contribueix significativament al temps total de processament del workflow.

Fitting: és una etapa que també requereix un temps considerable de computació. Aquest procés s'encarrega de l'ajustament dels models a les dades observades, treballant amb tot el conjunt de dades alhora. Això és especialment important per a l'estimació de punts de flux i corbes de llum.

Un cop analitzat els blocs, que més temps consumeixen, s'han detectat redundàncies en el bloc de Process Observations del workflow de SED mitjançant la modelització d'aquest pipeline, ja que es es processen i modifiquen els datasets corresponents a cada observació, i s'afegeixen a una col·lecció final dos cops. Un es fa internament amb la crida de la funció get_datasets() de la clase Analysis de la llibereia de Gammapy i l'altre es fa de manera manual amb un bucle.

Per dur a terme aquesta paral·lelització, és necessari trobar un entorn que ofereixi tots els recursos possibles per explorar la paral·lelització per a aquest projecte. Implementar aquestes solucions en un entorn com el Port d'Informació Científica (PIC), que proveeix plataformes de còmput distribuït per reduir els temps d'execució mitjançant tècniques de paral·lelisme, permet optimitzar aquests temps.

Donat el context, en el qual es desenvolupa el treball, l'objectiu principal d'aquest projecte consisteix a implementar una versió paral·lelitzada des de zero dels workflows d'anàlisi de la corba de llum i l'espectre de distribució d'energia pel bloc Process Observations, utilitzant la llibreria de Python, Dask. A més, es vol activar la paral·lelització desenvolupada pels creadors de Gammapy, pel bloc de Fitting, amb les llibreries de Python RAY i Multiprocessing. Tot i que aquestes llibreries són prototips i no estan completament funcionals, es vol provar les dues per determinar quina ofereix el millor rendiment i quedar-se amb la més eficient.

Dins d'aquest marc, es defineixen diversos subobjectius amb la intenció d'assolir una millora global en el rendiment i la gestió de dades:

1. **Desenvolupar workflows d'anàlisi generalitzats**: Això permetrà una major flexibilitat i reutilització dels recursos disponibles.
2. **Comparar el rendiment entre versions sèrie i paral·lelitzades**: Es realitzaran comparatives entre les versions sèrie i paral·lelitzada, com ara Dask-Sèrie, així com entre Multiprocessing i Ray, pels dos workflows.
3. **Establir mètriques**: Es volen establir mètriques per avaluar el temps total d'execució dels workflows, així com el temps d'execució per etapes específiques. També s'avaluaran la velocitat amb speedup, l'eficiència i el PI (Performance Index) que busca la millor relació entre la pèrdua d'eficiència i el guany en temps execució amb el valor mínim de l'índex.

Aquests subobjectius contribuiran a optimitzar el procés d'anàlisi de dades astrofísiques, maximitzant l'eficiència i la velocitat de processament en entorns de gran volum de dades.

4 DESENVOLUPAMENT

4.1 ENGINYERIA DEL RENDIMENT

Com bé s'ha comentat abans, s'han detectat redundàncies en el bloc de Process Observations del workflow de SED mitjançant la modelització d'aquest flux de treball, ja que es es processen i modifiquen els datasets corresponents a cada observació, i s'afegeixen a una col·lecció final dos cops.
Un es fa internament amb la crida de la funció get_datasets() de la clase Analysis de la llibreria de Gammapy i l'altre es fa de manera manual amb un bucle.
El que s'ha fet és eliminar la crida de la funció get_datasets(), per tal de deixar només el procés de les observacions que es faci de manera manual externament, per tal de poder paral·lelitzar posteriorment el bucle.

4.2 PARAL·LELITZACIÓ PROCESS OBSERVATIONS

Un cop s'ha aplicat l'enginyeria del rendiment sobre el Process Observations del workflow SED, es desenvolupa la paral·lelització distribuïda al bloc de Process Observations amb la llibreria Dask, tal com es mostra a la Figura 5.

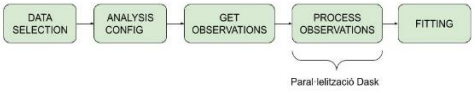

Figura 5: Etapa Process Observations

La raó principal de l'ús de Dask és que és l'objectiu del projecte PIC, que ha integrat una capa de Dask de nivell superior per permetre la paral·lelització distribuïda. Dask s'ha demostrat com una eina eficaç per gestionar grans volums de dades, oferint escalabilitat dinàmica i una millor utilització dels recursos disponibles.
Concretament, Dask es fa servir per paral·lelitzar el bucle mencionat anteriorment, el qual s'executa per cada observació astronòmica. En aquest bucle, es processen i modifiquen els conjunts de dades corresponents a cada observació, i s'afegeixen a una col·lecció final. Això implica distribuir aquest bucle de manera que N CPUs virtuals puguin processar simultàniament N observacions.

Per tal de desenvolupar i poder avaluar correctament la paral·lelització del bloc "Process Observations" mitjançant l'ús de Dask, s'han realitzat un conjunt d'experiments que permeten la reserva dinàmica de recursos de còmput per crear un clúster Dask. El clúster de Dask estarà format per N workers, on un worker és la unitat mínima de còmput que definim a la configuració. A partir de la configuració es detalla quantes CPUs, quanta memòria total i mida de disc volem per al nostre worker, i es replicarà tantes vegades com li indiquem en un segon valor d'escalabilitat i que ens dimensionarà la mida total del nostre clúster, tal com es mostra a la Figura 6.
Entès, aquí tens la frase amb l'addició:

```
cluster = HTCondorCluster(cores=1, memory='2GB',disk='2GB',job_extra_directives={'getenv': 'True'})
cluster.scale(16)
```

Figura 6: Creació interfície HTCondorCluster

En aquest cas, les combinacions utilitzades són scale=N i cores=1, on N canvia segons el nombre de CPUs necessàries. Aquesta configuració és la combinació perfecta per a la paral·lelització perquè cada scale es converteix en un slot independent que s'assigna a qualsevol lloc disponible en la màquina física. Això evita l'espera i la sobrecàrrega de comunicació, garantint un ús eficient dels recursos.

Mitjançant aquesta assignació Dask comunicarà la petició a HTCondor que reservarà CPUs virtuals allà on hi hagi recursos disponibles. Aquestes CPUs virtuals poden distribuir-se tant en diferents màquines físiques, cadascuna amb arquitectures heterogènies, com en la mateixa màquina física. Així, HTCondor pot gestionar més fàcilment aquestes CPUs virtuals i assignar-les on hi hagi disponibilitat, optimitzant l'ús dels recursos del clúster.
La paral·lelització s'ha realitzat en un format de laboratori utilitzant una arquitectura homogènia, la mateixa que en sèrie, per poder fer comparacions reals i amb les mateixes condicions. Aquest requisit s'ha especificat durant la creació del clúster Dask amb una primitiva de HTCondor. Aquesta metodologia ens ha permès garantir que els resultats obtinguts siguin fiables i que les diferències observades siguin degudes únicament a la paral·lelització i no a variacions en l'heterogeneïtat de les arquitectures del PIC.
Les proves s'han fet amb la següent arquitectura: AMD EPYC 7452 32-Core Processor, que és la millor disponible en aquest moment al PIC, tant pels components tecnològics i els avenços que aporta, si no també, per la ubicació d'aquestes màquines físiques, ja que es troben en la mateixa sala que el clúster de HTCondor i el disc en xarxa.

Arquitectura homogènia:
Els experiments han consistit a executar per cadascun dels workflows d'estudi d'aquest treball, la solució implementada en cada cas mitjançant la paral·lelització del bloc Process Observations utilitzant Dask, dimensionant el clúster que executarà la càrrega de feina a partir d'incrementar el número de CPUs i memòria, doblant els recursos a cada increment. De cada configuració se n'han fet un total de tres execucions per extreure'n un temps d'execució mig que es presenta a cada taula.
A més de les proves en format de laboratori, també s'han dut a terme proves en el format real, que inclou diferents arquitectures disponibles al PIC. Aquest enfocament ens ha permès ava-

luar el rendiment de la paral·lelització en un entorn més divers i representatiu de les condicions reals d'operació, on l'heterogeneïtat de les màquines físiques pot influir en els resultats i aportar soroll als resultats obtinguts.

Arquitectures heterogènies:
En aquest apartat es du a terme la prova real amb les diferents arquitectures disponibles al PIC:

- AMD EPYC 7452 32-Core Processor → màquines amb la nomenclatura: td8xx
- AMD EPYC 7502 32-Core Processor → màquines amb la nomenclatura: tds4xx

Totes dues tenen característiques tecnològiques molt similars, amb la diferència principal que la AMD EPYC 7502 32-Core Processor té una freqüència de rellotge de 2,5GHz, mentre que la AMD EPYC 7452 32-Core Processor funciona a 2,35GHz.
A més, la EPYC 7502 té 44 màquines físiques i 5632 slots (CPUs virtuals) disponibles, mentre que la EPYC 7452 té 32 màquines físiques i 4096 slots disponibles.

4.3 PARAL·LELITZACIÓ FITTING

Un cop desenvolupat la paral·lelització del bloc de Process Observations, toca desplegar la paral·lelització del bloc de Fitting, amb les llibreries RAY i Multiprocessing, tal com es mostra a la Figura 7.

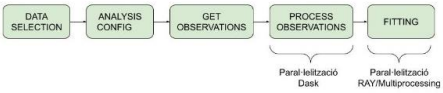

Figura 7: Etapa Fitting

La decisió de paral·lelitzar aquest bloc amb les llibreries RAY i Multiprocessing, es fonamenta en l'aprofitament de la paral·lelització integrada en les funcions específiques de Gammapy, "FluxPointsEstimator" i "LightCurveEstimator". Aquest enfocament permet fer ús de les capacitats de paral·lelització disponibles durant el bloc de Fitting, en lloc de considerar altres llibreries sense aquesta integració específica, representant així una oportunitat addicional d'optimització pels workflows. Aquesta paral·lelització, encara està en fase de desenvolupament, així doncs encara està en fase de prototip i no són totalment funcionals, ja que hi ha funcions i/o mètodes interns que encara no s'han paral·lelitzat.

La paral·lelització amb Multiprocessing fa referència a l'ús de múltiples threads d'una mateixa execució en un node de còmput per executar tasques de manera concurrent. Cada thread executa una part de la tasca de manera simultània, augmentant així la velocitat d'execució global.
En el cas de RAY, si bé permet la paral·lelització distribuïda, com que encara és un prototip, actualment no està configurat per a aquesta funcionalitat. Per tant, només és possible utilitzar-lo de manera no distribuïda en una única màquina física. En aquest escenari, RAY utilitza el concepte de multithreading per repartir la càrrega de computació entre múltiples fils d'execució. Per canviar d'una llibreria a una altra, es fa servir el paràmetre parallel_backend de les funcions esmentades anteriorment, on es pot especificar "ray" o "multiprocessing"

Els tests dels workflows s'han realitzat utilitzant 1, 2 CPUs, que ofereix Jupyter al PIC (que corre sobre el clúster HTCondor). Per a cada augment de CPU, es generen el mateix nombre de jobs per tal de dividir la càrrega de treball d'aquest bloc en N processos, assignant cada CPU a un d'aquests processos. En cada prova amb N CPUs i N jobs, també s'ha realitzat un test amb el doble de jobs respecte al nombre de CPUs per garantir una utilització completa dels recursos disponibles.
Tal com s'ha comentat anteriorment, el test per cada workflow es repeteix 3 vegades, i es fa la mitjana ponderada de les mètriques de temps.
Totes elles en la mateixa màquina física del PIC amb l'arquitectura següent:

- AMD EPYC 7452 32-Core Processor, identificada amb la nomenclatura: td8xx.

Corba de llum:
En aquest cas, només s'utilitza Multiprocessing, ja que RAY és encara un prototip no del tot operatiu, i encara no està disponible per aquest worflow.

SED:
En aquest cas, s'utilitzen les biblioteques RAY i Multiprocessing. Tot i ser prototips, totes dues són operatives, però alguns mètodes i funcions no contribueixen a estalviar temps en la implementació d'aquest anàlisi i permeten la paral·lelització.

5 RESULTATS

Les mètriques utilitzades per avaluar els resultats són les següents:

1. Speedup: Es calcula com el quocient entre el temps d'execució en sèrie i el temps d'execució en paral·lel. Aquesta mètrica ens indica com de més ràpid és el procés quan s'executa en paral·lel respecte a quan s'executa en sèrie.

$$Speedup = \frac{Temps\ sèrie}{Temps\ paral{\cdot}lelitzat}$$

2. Eficiència: Es calcula com el quocient entre el "Speedup" i el nombre de recursos (CPUs) utilitzats. Aquesta métrica és l'acceleració, indica com de bé s'estan utilitzant els recursos disponibles.

$$Eficiència = \frac{Speedup}{nº\ recursos}$$

3. PI: Es calcula com el quocient entre el temps d'execució en paral·lel i l'eficiència. Aquest índex indica la millor relació entre la pèrdua d'eficiència i el guany en execució. Aquesta mètrica és utilitzada per determinar el millor cas.

$$PI = \frac{Temps\ paral{\cdot}lelitzat}{eficiència}$$

5.1 RESULTATS ENGINYERIA DEL RENDIMENT

TEMPS (s) SED	
	SÈRIE
DATA SELECTION	0,15
ANALYSIS CONFIG	0,12
GET OBSERVATIONS	0,49
PROCESS OBSERVATIONS	340,48
FITTING	**921,55**
TOTAL CONJUNT TASQUES	1262,79

Taula 3: Etapa pre ER

TEMPS (s) SED	
	SÈRIE
DATA SELECTION	0,15
ANALYSIS CONFIG	0,12
GET OBSERVATIONS	0,49
PROCESS OBSERVATIONS	190,75
FITTING	**921,55**
TOTAL CONJUNT TASQUES	1113,06

Taula 4: Etapa post ER

Tal com es mostra a les Taules 3 i 4, un cop refactoritzar el codi, el temps s'ha reduït de 340,48 segons a 190,75 segons, aconseguint una acceleració, de 1,78x.

5.2 RESULTATS PROCESS OBSERVATIONS

Arquitectura homogènia

Cobra de llum:

MATEIXA ARQUITECTURA					
CORBA DE LLUM - TEMPS (s)					
	1 CPU	2 CPUs	4 CPUs	8 CPUs	16 CPUs
PROCESS OBSERVATION	125,34	74,85	41,57	23,72	14,30

Taula 5: Temps arquitectura homogènia Corba llum

Tal com es mostra a la Taula 5, el rendiment de la paral·lelització del workflow, és propera a la ideal. Per cada duplicació dels recursos, el temps d'execució disminueix aproximadament a la meitat. Aquest comportament demostra que l'algoritme de paral·lelització està funcionant de manera eficient i escalant correctament amb l'augment dels recursos disponibles.
A continuació, analitzem les diferents mètriques de rendiment per aquesta configuració.

MATEIXA ARQUITECTURA				
PROCESS OBSERVATIONS- CORBA DE LLUM				
	1 CPU- 2 CPUs	1 CPU- 4 CPUs	1 CPU- 8 CPUs	1 CPU- 16 CPUs
SPEEDUP	1,67	3,02	5,28	8,76
EFICIÈNCIA	0,84	0,76	0,66	0,55
PI	89,11	54,69	35,93	26,11

Taula 6: Mètriques arquitectura homogènia Corba llum

La Taula 6, presenta les mètriques de Speedup, Eficiència i PI per a les diferents proves de paral·lelització que s'han fet. L'acceleració idealment hauria de ser igual al nombre de CPUs utilitzades i idealment, l'eficiència hauria de ser 1, indicant que cada CPU addicional contribueix plenament al rendiment.

Tanmateix, com que l'acceleració no és ideal, tampoc l'eficiència s'apropa a 1. La disminució de l'acceleració i l'eficiència amb l'augment del nombre de CPUs reflecteix la sobrecàrrega de gestió de tasques i la competència pels recursos. Això indica que el rendiment millora amb l'augment del nombre de CPUs, però no de manera proporcional.
A més, hi ha una tendència de saturació a partir de 8 CPUs, per això el PI comença a disminuir, fent que, si es passés a 32 CPUs, els guanys serien mínims.

El millor cas és el de la paral·lelització amb 16 CPUs, ja que té el valor més baix de PI, tal com es mostra a la Figura 8. Això indica que aquesta configuració ofereix el millor rendiment global, tenint en compte tant el temps d'execució en paral·lel com l'eficiència d'ús dels recursos.

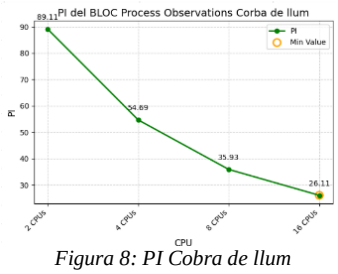

Figura 8: PI Cobra de llum

SED:

MATEIXA ARQUITECTURA					
SED - TEMPS (s)					
	1 CPU	2 CPUs	4 CPUs	8 CPUs	16 CPUs
PROCESS OBSERVATION	190,75	91,05	47,47	26,69	14,36

Taula 7: Temps arquitectura homogènia SED

A la Taula 7, la paral·lelització segueix el mateix patró: per cada duplicació dels recursos, el temps d'execució es redueix aproximadament a la meitat.
A continuació, analitzem les diferents mètriques de rendiment per aquesta configuració.

MATEIXA ARQUITECTURA				
PROCESS OBSERVATIONS - SED				
	1 CPU- 2 CPUs	1 CPU- 4 CPUs	1 CPU- 8 CPUs	1 CPU- 16 CPUs
SPEEDUP	2,09	4,02	7,14	13,32
EFICIÈNCIA	1,04	1,01	0,89	0,83
PI	87,54	47,00	29,98	17,30

Taula 8: Mètriques arquitectura homogènia SED

En els resultats de la Taula 8, l'acceleració ha aconseguit el cas ideal per 2 CPUs i per a 4 CPUs, mentre que amb 8 CPUs i 16 CPUs s'ha apropat al cas ideal. Aquesta variació mostra una eficiència notablement alta en els casos de 2 i 4 CPUs, ja que el l'acceleració és igual al nombre de CPUs, indicant una distribució de càrrega ideal i un aprofitament complet dels recursos disponibles.
Per contra, amb 8 i 16 CPUs, tot i que l'acceleració és alta, no arriba al cas ideal. Això és degut a factors com la sobrecàrrega i la latència en la comunicació entre les CPUs.
L'eficiència obtinguda és força elevada i segueix el mateix patró que l'acceleració. Aquest nivell d'eficiència indica una utilització molt efectiva de les CPUs, malgrat les limitacions mencionades anteriorment, mantenint la mateixa tendència de saturació.

El millor rendiment es troba amb la paral·lelització utilitzant 16 CPUs, tal es mostra a la Figura 9 de l'apèndix A1. Aquesta configuració demostra el millor temps d'execució en paral·lel i una eficiència notable en l'ús dels recursos.

Arquitectura heterogènia

Corba de llum:

DIFERENT ARQUITECTURA					
CORBA DE LLUM - TEMPS (s)					
	1 CPU	2 CPUs	4 CPUs	8 CPUs	16 CPUs
PROCESS OBSERVATION	125,34	100,42	55,03	28,25	17,30

Taula 9: Temps diferent arquitectura Corba llum

A la Taula 9, l'ús d'arquitectures heterogènies mitiguen la paral·lelització. Malgrat això, el temps d'execució disminueix aproximadament a la meitat per cada duplicació de recursos, indicant una eficiència propera a l'ideal. Tot i això, els resultats no igualen els de l'arquitectura homogènia, demostrant que l'algoritme de paral·lelització escala eficientment amb més recursos, però amb una penalització quan s'utilitzen arquitectures heterogènies.

Analitzem també la distribució de tasques entre les arquitectures heterogènies del PIC, la qual pot limitar l'execució. A continuació es mostren alguns comptes de les execucions:

2 CPUs	4 CPUs
td828.pic.es: 79 tds422.pic.es: 76	td817.pic.es: 38 tds416.pic.es: 39 td819.pic.es: 37 td828.pic.es: 41
8 CPUs	**16 CPUs**
tds426.pic.es: 24 tds418.pic.es: 13 td826.pic.es: 25 tds416.pic.es: 23 tds414.pic.es: 19 td827.pic.es: 6 tds415.pic.es: 19 tds423.pic.es: 26	tds409.pic.es: 12 tds402.pic.es: 1 td826.pic.es: 13 tds441.pic.es: 14 tds432.pic.es: 11 tds440.pic.es: 2 tds416.pic.es: 10 tds415.pic.es: 2 td819.pic.es: 19 tds423.pic.es: 15 td815.pic.es: 10 td817.pic.es: 14 tds426.pic.es: 13 td827.pic.es: 13 tds403.pic.es: 5 tds414.pic.es: 1

Taula 10: Tasques Corba llum

A la Taula 10, s'observa que l'arquitectura EPYC 7502 és àmpliament utilitzada i la més abundant al PIC, amb 56 màquines físiques i 5632 slots disponibles. Tanmateix, la seva eficiència real és limitada per la infraestructura de comunicacions. Les màquines estan submergides i connectades a través de patch panels, una fibra de 60m, switchs i uplinks amb restriccions a 1Gbps, la qual cosa afecta l'amplada de banda i incrementa el temps d'execució a causa de la sobrecàrrega de comunicació.

Per contra, les màquines no submergides, situades en una sala amb connexió a 10Gbps i accés directe als switchs amb fibres curtes, ofereixen millors condicions de xarxa. Encara que són tecnològicament més antigues, la seva ubicació prop del disc en xarxa amb les 155 observacions i els servidors d'HTCondor, i l'amplada de banda superior, eviten aquesta sobrecàrrega de comunicació.

Per corroborar aquest raonament, s'ha realitzat un test: s'ha definit un servidor d'iperf en una màquina del PIC i s'ha establert un client iperf en un node tds4XX i un td8XX per veure el rendiment de la xarxa en transferir amb 50 connexions, tal com es mostra a les Figures 10 i 11.

```
[jdelgado@tds404 tmp]$ iperf -c 193.109.175.21 -P 50 -p 5001 -f g -t 5
[ 40] local 192.168.101.114 port 43612 connected with 193.109.175.21 port 5001
[ 46] local 192.168.101.114 port 43792 connected with 193.109.175.21 port 5001
[ 45] local 192.168.101.114 port 43774 connected with 193.109.175.21 port 5001
[ 21] local 192.168.101.114 port 43538 connected with 193.109.175.21 port 5001
[ 27] local 192.168.101.114 port 43580 connected with 193.109.175.21 port 5001
[ 32] local 192.168.101.114 port 43688 connected with 193.109.175.21 port 5001
[  6] 0.00-5.35 sec  0.010 GBytes  0.016 Gbits/sec
[ 45] 0.00-5.36 sec  0.032 GBytes  0.051 Gbits/sec
[SUM] 0.00-5.13 sec  1.19 GBytes  1.99 Gbits/sec
```

Figura 10: Prova amb la màquina física tds404 amb l'arquitectura: AMD EPYC 7502 32-Core Processor

```
[jdelgado@td831 ~]$ iperf -c 193.109.175.21 -P 50 -p 5001 -f g -t 5
[ 21] local 192.168.102.41 port 32790 connected with 193.109.175.21 port 5001
[ 10] local 192.168.102.41 port 32912 connected with 193.109.175.21 port 5001
[  1] local 192.168.102.41 port 60984 connected with 193.109.175.21 port 5001
[ 34] local 192.168.102.41 port 32770 connected with 193.109.175.21 port 5001
[ 16] local 192.168.102.41 port 32782 connected with 193.109.175.21 port 5001
[ 19] local 192.168.102.41 port 33032 connected with 193.109.175.21 port 5001
[ 43] local 192.168.102.41 port 33016 connected with 193.109.175.21 port 5001
[ 40] local 192.168.102.41 port 33020 connected with 193.109.175.21 port 5001
[ 41] 0.00-5.04 sec  0.224 GBytes  0.382 Gbits/sec
[ 32] 0.00-5.05 sec  0.241 GBytes  0.410 Gbits/sec
[ 45] 0.00-5.07 sec  0.321 GBytes  0.543 Gbits/sec
[SUM] 0.00-5.05 sec  14.2 GBytes  24.1 Gbits/sec
```

Figura 11: Prova amb la màquina física tds404 amb l'arquitectura: AMD EPYC 7452 32-Core Processor

Amb la mateixa configuració de client, els nodes submergits han enviat 1,19 GB amb un throughput de 1,99 Gbits/seg, mentre que els nodes no submergits han enviat 14,2 GB amb un throughput de 24,1 Gbits/seg. Això mostra que els nodes no submergits, amb una amplada de banda superior, tenen millor comunicació i menys sobrecàrrega.

A continuació, analitzem les diferents mètriques de rendiment per aquesta configuració.

DIFERENT ARQUITECTURA PROCESS OBSERVATIONS- CORBA DE LLUM				
	1 CPU- 2 CPUs	1 CPU- 4 CPUs	1 CPU- 8 CPUs	1 CPU- 16 CPUs
SPEEDUP	1,24	2,27	4,44	7,25
EFICIÈNCIA	0,62	0,57	0,55	0,45
PI	161,96	96,96	50,90	38,17

Taula 11: Mètriques arquitectura heterogènia Corba llum

Segons les mètriques de la Taula 11, la sobrecàrrega de comunicació impedeix una acceleració i eficiència ideals, seguint una tendència de saturació. El millor rendiment es troba amb la paral·lelització utilitzant 16 CPUs, amb el valor de PI més baix, com es mostra a la Figura 12 de l'apèndix A1, demostrant la millor execució en paral·lel i eficiència de recursos.

SED:

DIFERENT ARQUITECTURA SED - TEMPS (s)					
	1 CPU	2 CPUs	4 CPUs	8 CPUs	16 CPUs
PROCESS OBSERVATION	190,75	109,03	55,93	28,85	16,70

Taula 12: Temps arquitectura heterogènia SED

A la Taula 12, es mostra el mateix patró que anteriorment, l'arquitectura heterogènia mitiga la paral·lelització. Aquesta penalització és degut al fet que hi ha una alta presència de l'arquitectura EPYC 7502, tal com s'indica a la Taula 13 de l'apèndix A1, i com que són màquines submergides es presenta el mateix problema de comunicacions descrit anteriorment.

A continuació, analitzem les diferents mètriques de rendiment per aquesta configuració.

DIFERENT ARQUITECTURA PROCESS OBSERVATIONS - SED				
	1 CPU- 2 CPUs	1 CPU- 4 CPUs	1 CPU- 8 CPUs	1 CPU- 16 CPUs
SPEEDUP	1,75	3,41	6,61	11,42
EFICIÈNCIA	0,87	0,85	0,82	0,71
PI	125,32	65,80	35,18	23,21

Taula 14: Mètriques arquitectura heterogènia SED

Tal com es mostra a les mètriques de la Taula 14, la sobrecàrrega de comunicació impedeix una acceleració i eficiència ideals, seguint la mateixa tendència de saturació. El millor rendiment es troba amb la paral·lelització utilitzant 16 CPUs, com es veu a la Figura 13 de l'apèndix A1.

5.3 RESULTATS FITTING

Corba de llum:

CORBA DE LLUM - TEMPS (s)					
	1 CPU	2 CPUs 2 Jobs	2 CPUs 4 Jobs	4 CPUs 4 Jobs	4 CPUs 8 Jobs
FITTING	62,01	44,60	37,00	34,50	33,29

Taula 15: Temps Fitting Corba llum

FITTING - CORBA DE LLUM				
	2 CPUs 2 Jobs	2 CPUs 4 Jobs	4 CPUs 4 Jobs	4 CPUs 8 Jobs
SPEEDUP	1,39	1,67	1,79	1,86
EFICIÈNCIA	0,70	0,84	0,45	0,46
PI	63,71	44,04	76,66	72,36

Taula 16: Mètriques Fitting Corba llum

La millora de temps observada en els resultats de la paral·lelització no és significativa, com es pot veure a les Taules 15 i 16. Això podria ser atribuït al fet que la paral·lelització està en una fase inicial de desenvolupament, amb funcions i mètodes que encara no s'han optimitzat completament per a la paral·lelització. És possible que encara hi hagi parts del codi que no es puguin executar de manera paral·lela a causa de les limitacions en el desenvolupament del prototip.
Un dels factors clau que afecten aquest resultat és que, malgrat utilitzar múltiples CPUs, la càrrega de treball no aconsegueix utilitzar el 100% dels recursos disponibles de manera eficient. Això es reflecteix en els valors d'eficiència observats a les taules, que són menors que l'ideal d'1.
També es reflecteix en la mètrica de l'acceleració, ja que és significativament menor a l'ideal. Això mostra que el rendiment no millora de manera proporcional a l'augment del nombre de CPUs.
Tot i això, en el segon cas, en posar el doble de càrrega amb 2 CPUs i 4 jobs, es redueix significativament el temps d'execució. Aquesta millora indica que en aquest escenari és possible que aprofiti el 100% de les CPUs disponibles. No obstant això, quan augmentem el nombre de CPUs i doblem la càrrega a 4 CPUs i 8 jobs, la millora és mínima.
El millor cas es troba és el de la paral·lelització utilitzant 2 CPUs i 4 jobs, basant-se en la mètrica PI, tal com es mostra a la Figura 14.

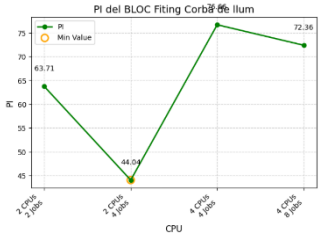

Figura 14: PI Fitting Corba llum

SED Multiprocessing:

SED MULTIPROCESSING - TEMPS (s)					
	1 CPU	2 CPUs 2 Jobs	2 CPUs 4 Jobs	4 CPUs 4 Jobs	4 CPUs 8 Jobs
FITTING	921,55	659,26	422,66	380,48	370,00

Taula 17: Temps Fitting SED Multiprocessing

FITTING - SED MUTLIPROCESSING				
	2 CPUs 2 Jobs	2 CPUs 4 Jobs	4 CPUs 4 Jobs	4 CPUs 8 Jobs
SPEEDUP	1,39	2,18	2,42	2,49
EFICIÈNCIA	0,70	1,09	0,60	0,62
PI	941,80	387,76	628,89	594,37

Taula 18: Mètriques Fitting SED Multiprocessing

Tal com es mostra a les Taules 17 i 18, s'observa un patró en què la millora de temps amb la paral·lelització no és molt significativa. A partir de la configuració de 2 CPUs amb 4 tasques, la millora és mínima, mantenint l'eficiència i l'acceleració més o menys constants.
El millor cas es troba en la configuració de 2 CPUs amb 4 tasques, ja que en aquest escenari, la càrrega de treball per CPU permet aprofitar eficientment el 100% de les CPUs disponibles, tot i les limitacions del sistema, perquè encara està en fase de desenvolupament i optimització, fent que encara hi hagi funcions i/o mètodes sense la possibilitat de paral·lelitzar. Aquesta configuració té el valor més baix de PI, com es mostra a la Figura 15 de l'apèndix A2.

SED RAY:

SED RAY - TEMPS (s)					
	1 CPU	2 CPUs 2 Jobs	2 CPUs 4 Jobs	4 CPUs 4 Jobs	4 CPUs 8 Jobs
FITTING	921,55	714,61	433,03	393,95	375,29

Taula 19: Temps Fitting SED RAY

FITTING - SED RAY				
	2 CPUs 2 Jobs	2 CPUs 4 Jobs	4 CPUs 4 Jobs	4 CPUs 8 Jobs
SPEEDUP	1,29	2,12	2,33	2,45
EFICIÈNCIA	0,64	1,06	0,58	0,61
PI	1116,57	408,51	679,22	615,22

Taula 20: Mètriques Fitting SED RAY

Aquesta prova de paral·lelització amb RAY segueix el mateix patró que l'anterior cas amb Multiprocessing, a partir de la configuració de 2 CPUs i 4 Jobs, les millores són mínimes, per les mateixes raons descrites anteriorment.
Així doncs, el millor cas és la configuració de 2 CPUs amb 4 jobs, ja que demostra la millor relació entre la pèrdua d'eficiència i el guany en execució, tal com es mostra a la Figura 15, que es troba l'apèndix A2.

D'altra banda, s'observa que Multiprocessing ofereix millors resultats, és a dir, és més eficient que RAY. Això es deu al fet que RAY a més de tenir la limitació perquè encara és un prototip i hi ha funcions i/o mètodes que encara no es poden paral·lelitzar, està configurat per permetre la paral·lelització distribuïda, la qual cosa introdueix una configuració més complexa i, per tant, genera una sobrecàrrega addicional.

A continuació, es mostra els nous temps dels worfklows SED i Corba de llum, aplicant les configuracions de paral·lelització.

TEMPS (s) SED		
	SÈRIE	PARAL·LEL
DATA SELECTION	0,15	0,16
ANALYSIS CONFIG	0,12	0,12
GET OBSERVATIONS	0,49	0,57
PROCESS OBSERVATIONS	190,75	17,20
FITTING	**921,55**	**421,93**
TOTAL CONJUNT TASQUES	1113,06	439,98

Taula 21: SED final

TEMPS (s) CORBA LLUM		
	SÈRIE	PARAL·LEL
DATA SELECTION	2,73	2,55
ANALYSIS CONFIG	0,01	0,01
GET OBSERVATIONS	0,6	0,5
PROCESS OBSERVATIONS	**125,34**	**16,93**
FITTING	**62,01**	**37,31**
TOTAL CONJUNT TASQUES	190,69	57,30

Taula 22: Corba llum final

Tal com es mostra a les Taules 21 i 22, en el cas del workflow final del SED, s'ha aconseguit una acceleració final de 2,52x amb les següents configuracions:

- Process Observations utilitzant 16 CPUs.
- Fitting amb 2 CPUs i 4 jobs en paral·lel.

Pel que fa a la Corba de llum, s'ha obtingut una acceleració final de 3,32x en comparació amb l'execució en sèrie, amb la següent configuració:

- Process Observations també utilitzant 16 CPUs.
- Fitting amb 2 CPUs i 4 jobs en paral·lel.

Aquestes configuracions han demostrat ser efectives per reduir significativament el temps d'execució total, mostrant una adaptació eficaç de la paral·lelització a les diferents tasques d'anàlisi.

6 CONCLUSIONS

Es pot concloure que tots els objectius inicials d'aquest projecte s'han assolit amb èxit. Principalment, s'ha completat la implementació des de zero dels workflows d'anàlisi de la corba de llum i l'espectre de distribució d'energia, utilitzant la llibreria Python Dask per al bloc de Process Observations. A més, s'ha aconseguit activar eficientment la paral·lelització del bloc de Fitting amb les llibreries Python RAY i Multiprocessing, desenvolupades pels creadors de Gammapy.

Això demostra que s'ha abordat el problema amb eficiència i adequació, permetent gestionar grans volums de dades sense complicacions significatives. Gràcies a la implementació de la paral·lelització, ara és possible manejar i escalabilitat sense problemes conjunts de dades que abans es limitaven a GB, ja que els workflows paral·lelitzats tenen la capacitat de treballar eficaçment amb TB de dades.

En segon lloc, s'ha enfocat a desenvolupar workflows generalitzats per a proporcionar una estructura adaptable i flexible per a diverses necessitats. A més, s'ha dut a terme un anàlisi exhaustiu comparant el rendiment entre les versions paral·lelitzades i les seqüencials, així com les paral·lelitzacions entre Multiprocessing i RAY.

Amb Dask, la paral·lelització distribuïda mostra un rendiment proper a l'ideal quan s'executa en la mateixa màquina, minimitzant les heterogeneïtats introduïdes pel PIC. Per cada duplicació dels recursos, el temps d'execució es redueix aproximadament a la meitat, demostrant l'eficàcia de l'algoritme de paral·lelització i la seva capacitat d'escalabilitat amb l'increment dels recursos disponibles. Tot i això, és important assenyalar que, en realitat, l'heterogeneïtat no s'elimina completament. Malgrat això, Dask continua paral·lelitzant de manera eficient, encara que no s'aproxima al cas ideal.

L'activació de la paral·lelització de les llibreries RAY i Multiprocessing, han jugat un paper clau en la millora del rendiment dels workflows. No obstant això, s'ha observat que Multiprocessing ha estat més eficient en la reducció del temps d'execució del bloc Fitting en comparació amb RAY, que introdueix una configuració més complexa i, per tant, una sobrecàrrega addicional.

S'ha de tenir en compte que encara pot haver-hi parts del codi que no es puguin executar de manera paral·lela, donada la fase de desenvolupament i optimització del prototip actual.

Finalment, com a línies futures, es contempla la comparació amb la versió de paral·lelització ja desenvolupada pel PIC utilitzant Spark, juntament amb una anàlisi comparativa amb la implementació actual de Dask. Aquesta avaluació proporcionarà una visió exhaustiva del rendiment de les diverses opcions disponibles.

Addicionalment, es preveu realitzar proves amb conjunts de dades de gran volum per validar l'escalabilitat del sistema en escenaris de càrrega màxima.

Per concloure, es proposa implementar la paral·lelització completa de tots els blocs dels workflows amb l'extracció de dades directament des de la base de dades Hive, millorant així l'eficiència i la gestió dels recursos.

BIBLIOGRAFIA

[1] Wikipedia. (2024). Astrofísica.
https://ca.wikipedia.org/wiki/Astrof%C3%ADsica
[2]Wikipedia. (2024). Astronomía.
https://ca.wikipedia.org/wiki/Astronomia
[3]Khan Academy. (2016). Light and the Electromagnetic Spectrum.
https://acortar.link/uVvtL
[4]Quimica.es. (2022). Rayos gamma
https://n9.cl/ei389
[5] Wikipedia. (2024). Radiació de Cherenkov.
https://n9.cl/777wh
[6] Wikipedia. (2021). Telescopio MAGIC.
https://es.wikipedia.org/wiki/Telescopio_MAGIC
[7]CTA. (2023). Cherenkov Telescope Array.
https://shorturl.at/0FMn8
[8] Gammapy. (2023).
https://gammapy.org/
[9] J. Delgado, A. Bruzzese, C. Nigro, J. Rico, G. Merino P. Tallada, J. Casals, J.
Carretero. (2022). GammaHub prototype [PowerPoint].
[10] Gammapy. (2023). Analysis Time - Light Curve.
https://shorturl.at/24kPd
[11] Gammapy. (2023). Analysis 1D - Spectral Analysis.
https://shorturl.at/BiFAB
[12] Port d'Informació Científica (PIC)
https://shorturl.at/0zJ3l
[13]Dask. (2022).
https://www.dask.org/
[14] HTCondor.(2024).
https://en.wikipedia.org/wiki/HTCondor
[15] RAY (2024).
https://shorturl.at/tTFsd
[16] Multiprocessing (2024).
https://docs.python.org/3/library/multiprocessing.html
[17] Graffica. (2016). FITS: Formato de archivo.
https://graffica.info/fits-formato-de-archivo/
[18] TCSPC. (2023). instrument response function (IRF).
https://n9.cl/fu647
[19]Astropy. (2024). HDUs.
https://docs.astropy.org/en/stable/io/fits/api/hdus.html

AGRAÏMENTS

Un profund agraïment i reconeixement dirigits a l'Eduardo César Galobardes, qui ha exercit un paper crucial com a professor guia del meu TFG, proporcionant una orientació valuosa i un suport constant durant tot el procés d'elaboració d'aquest treball.

A més, vull expressar el meu agraïment sincer a en Jordi Delgado, que també ha estat un tutor essencial, aportant la seva experiència i orientació per a la realització d'aquest projecte.

APÈNDIX

A1. SECCIÓ D'APÈNDIX

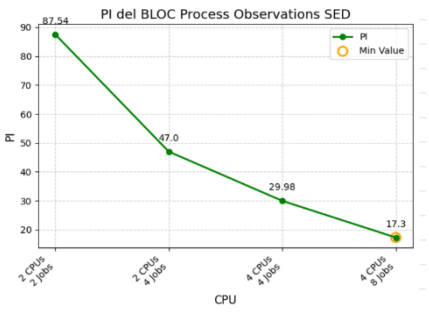

Figura 9: Gràfica PI SED arquitectura Homogènia

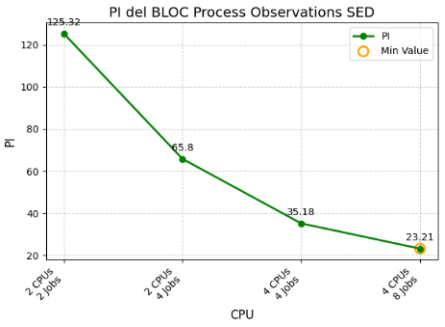

Figura 13: Gràfica PI SED arquitectura Heterogènia

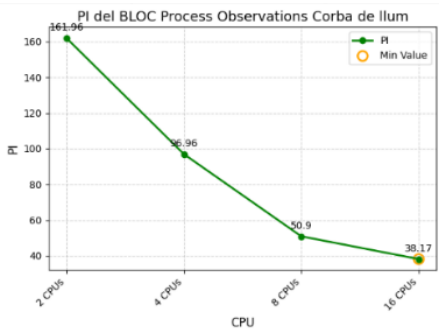

Figura 14: Gràfica PI Corba llum arquietctura Heterogènia

2 CPUs	4 CPUs
tds404.pic.es: 79 td813.pic.es: 76	tds404.pic.es: 40 tds439.pic.es: 38 td827.pic.es: 35 tds434.pic.es: 42
8 CPUs	**16 CPUs**
tds410.pic.es: 12 tds439.pic.es: 22 tds401.pic.es: 24 td820.pic.es: 20 td827.pic.es: 22 tds441.pic.es: 19 tds434.pic.es: 25 tds404.pic.es: 11	tds423.pic.es: 11 tds434.pic.es: 11 tds414.pic.es: 8 gpu05.pic.es: 21 tds410.pic.es: 6 td821.pic.es: 13 td827.pic.es: 10 tds415.pic.es: 11 td820.pic.es: 12 tds441.pic.es: 10 tds405.pic.es: 8 tds404.pic.es: 8 tds439.pic.es: 6 tds401.pic.es: 8 tds406.pic.es: 5 tds403.pic.es: 7

Taula 13: Taula tasques SED

A2. SECCIÓ APÈNDIX

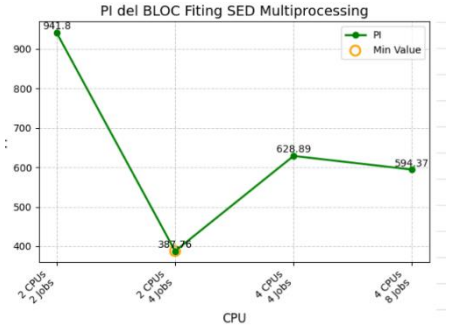

Figura 15: Gràfica PI SED Multiprocessing

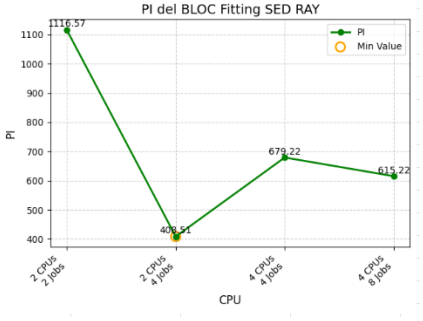

Figura 16: Gràfica PI SED RAY