



UNIVERSIDAD
DE LA REPUBLICA
URUGUAY



Computación de alto desempeño en plataformas cloud para la detección de rayos cósmicos en imágenes de telescopio

Germán Schnyder

Programa de Posgrado en Ingeniería en Informática
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
Setiembre de 2017



UNIVERSIDAD
DE LA REPUBLICA
URUGUAY



Computación de alto desempeño en plataformas cloud para la detección de rayos cósmicos en imágenes de telescopio

Germán Schnyder

Tesis de Maestría presentada al Programa de Posgrado en Ingeniería en Informática, Facultad de Ingeniería de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Magíster en Ingeniería en Informática.

Directores:

Ph.D. Prof. Sergio Nesmachnow
D.Sc. Prof. Gonzalo Tancredi

Director académico:

Ph.D. Prof. Sergio Nesmachnow

Montevideo – Uruguay

Setiembre de 2017

Schnyder, Germán

Computación de alto desempeño en plataformas cloud para la detección de rayos cósmicos en imágenes de telescopio / Germán Schnyder. - Montevideo: Universidad de la República, Facultad de Ingeniería, 2017.

XIV, 82 p.: il.; 29, 7cm.

Directores:

Sergio Nesmachnow

Gonzalo Tancredi

Director académico:

Sergio Nesmachnow

Tesis de Maestría – Universidad de la República, Programa en Ingeniería en Informática, 2017.

Referencias bibliográficas: p. 78 – 82.

1. Cloud Computing,
 2. HPC,
 3. Rayos Cómicos,
 4. Hubble,
 5. MapReduce,
 6. Hadoop,
 7. Mesos,
 8. Azure,
 9. Astronomía.
- I. Nesmachnow, Sergio, Tancredi, Gonzalo, . II. Universidad de la República, Programa de Posgrado en Ingeniería en Informática.
- III. Título.

INTEGRANTES DEL TRIBUNAL DE DEFENSA DE TESIS

Ph.D. Prof. NOMBRE APELLIDO

Ph.D. Prof. NOMBRE APELLIDO

D.Sc. Prof. NOMBRE APELLIDO

Montevideo – Uruguay
Setiembre de 2017

Agradecimientos

El trabajo de construir software dedicado al procesamiento de imágenes astronómicas surge como una inquietud personal de relacionar mi carrera universitaria (y profesional) con uno de mis mayores intereses: el espacio exterior. Este planteo fue entendido de inmediato por quien sería mi tutor para esta tesis, quien comenzó a mover los engranajes para ponerme en contacto con gente especializada en el área que me permitiera encontrar desafíos acordes a lo que se espera de una tesis de maestría pero sobre todo a los míos propios. En este sentido, comienzo por agradecer a Sergio Nesmachnow, director académico y co-tutor de esta tesis quien prestó especial atención a todos los detalles y se armó de paciencia para revisar cada aspecto metodológico y de redacción de este trabajo.

En segundo lugar, mi agradecimiento para los expertos en física y astronomía que recibieron mis consultas y me proveyeron de las herramientas para entender el dominio del problema: Gonzalo Tancredi, co-tutor de esta tesis y artífice de la idea original de utilizar los instrumentos del HST para la detección de los rayos cósmicos, proyecto que llevó a cabo con Santiago Roland, quien me cedió el código del prototipo original escrito en CL así como un conjunto inicial de imágenes para comenzar a realizar las primeras pruebas.

Las imágenes del HST fueron obtenidas a través de la gestión de Susana Deustua, científica del Space Telescope Science Institute (STScI) quien facilitó los discos con la totalidad de las imágenes del período operativo del HST y cuya colaboración fue resultado de la generosa intermediación de Gonzalo.

Para futuras investigaciones en base a este trabajo, el STScI otorgó una beca económica que de alguna manera valida la utilidad de los resultados obtenidos. Esta beca será aprovechada por estudiantes de doctorado en astronomía en tiempos venideros. Los detalles pueden encontrarse en <http://www.stsci.edu/institute/grants>.

Respecto a cuestiones técnicas, la infraestructura necesaria para ejecutar

el código de análisis de las imágenes fue provisto por Microsoft a través de su programa “Microsoft Azure Sponsorship” cuyos detalles pueden encontrarse en <https://azure.microsoft.com/en-us/offers/ms-azr-0036p/>.

Antes de pasar a la ejecución en Azure, utilicé la infraestructura de la Facultad de Ingeniería, y en particular el cluster del grupo de cálculo CeCal, así como el servidor Nebula. En este sentido, va mi agradecimiento en particular a Miguel Da Silva por su paciencia para resolver problemas de conexión de discos e instalación y configuración de software requerido para las ejecuciones.

El código fuente desarrollado para esta tesis se encuentra disponible bajo licencia MIT en la dirección <https://github.com/germanschnyder/cosmicrays/>. Para el desarrollo se utilizó PyCharm, con una licencia provista por JetBrains con fines académicos (<https://www.jetbrains.com/student/>).

Finalmente, a todos aquellos que forman parte de mi vida diaria, especialmente a Adriana y a todos los que de una forma u otra ayudaron a mantener mi motivación en alto.

COMPUTACIÓN DE ALTO DESEMPEÑO

EN PLATAFORMAS CLOUD PARA LA DETECCIÓN DE RAYOS CÓSMICOS EN IMÁGENES DE TELESCOPIO

RESUMEN

Los rayos cósmicos son partículas cargadas de energía y por su naturaleza son especialmente dañinos para los dispositivos electrónicos. Este tipo de rayos pueden ser medidos por instrumentos a nivel terrestre, excepto en algunas regiones debido a anomalías magnéticas en la atmósfera. Para estos casos se pueden utilizar mediciones de dispositivos que están por fuera de la atmósfera, como el Telescopio Espacial Hubble (HST). Los instrumentos a bordo del HST realizan tareas de mantenimiento regulares, que implican obtener imágenes con el lente tapado denominadas *darks*. Estas imágenes registran el impacto de los rayos cósmicos como consecuencia de la radiación cósmica. Se propone analizar el conjunto total de imágenes de tipo dark provisto por el Space Telescope Science Institute (STScI) correspondientes a los instrumentos del HST para estudiar el estado actual del campo magnético de la Tierra. En este trabajo se presentan tres alternativas para construir una arquitectura que permite procesar los más de 15 TB de información en formato de imágenes: MapReduce clásico con Hadoop, una solución combinando tecnologías del ecosistema de Apache Mesos y una arquitectura diseñada específicamente para correr en el ecosistema de Microsoft Azure. El punto

de partida de este trabajo es un prototipo desarrollado por la Facultad de Ciencias para limpiar las imágenes que se encuentran en formato IRAF (Image Reduction and Analysis Facility) y obtener la información acerca de los rayos cósmicos en diferentes posicionamientos de los instrumentos del HST. El resultado final de esta tesis supera en 20 veces el tamaño, en términos de volumen de datos, al prototipo inicial. La contribución principal de este trabajo es la descripción e implementación de una arquitectura paralela que permite acelerar los cálculos de forma drástica, con respecto a la aproximación original al problema.

Palabras clave: Cloud Computing, HPC, Rayos Cósmicos, Hubble, MapReduce, Hadoop, Mesos, Azure, Astronomía

Listado de figuras

1.1	Magnetósfera de la Tierra (imagen de dominio público, crédito NASA/Goddard/Aaron Kaase)	2
1.2	Anomalía del Atlántico sur (imagen de dominio público, tomada de Snowden (2011))	3
2.1	Imagen dark con impactos de rayos cósmicos (sección de j6mf16lhq.raw.fits)	12
2.2	Período operativo de cada instrumento del HST (imagen de dominio público, tomada de Wikipedia)	13
2.3	Captura de la luz por parte del HST (imagen de dominio público, tomada de Space Telescope Science Institute, 2017a)	14
2.4	Disposición física de los instrumentos del HST (imagen de dominio público, tomada de Riley y Biretta, 2017)	14
2.5	Estructura de la WFCP2 (imagen de dominio público, tomada de McMaster y Biretta, 2008)	17
2.6	Imágenes obtenidas por la WFPC2 [imagen de dominio público, crédito: NASA/ESA/STScI/Arizona State University]	18
2.7	Estructura de imagen FITS con extensiones (imagen de dominio público, tomada de Smith et al., 2011)	19
2.8	Formato de imagen de UVIS con todos los CCD incluidos (imagen de dominio público, tomada de Dressel, 2017)	21
3.1	Modelos de servicio cloud organizados como capas en una pila (imagen de dominio público tomada de Wikipedia y traducida al español)	27
3.2	Aplicación del paralelismo a nivel de datos al procesamiento de archivos	30

3.3	Aplicación del paralelismo a nivel de instrucción al procesamiento de una imagen	31
3.4	Arquitectura de tipo master-slave para la replicación de datos	33
3.5	Arquitectura de tipo cluster con forma de anillo para la replicación de los datos	34
4.1	Arquitectura de la solución (imagen con permiso de reproducción, tomada de Li et al. (2010) y traducida al español)	41
4.2	Diseño del planificador (imagen con permiso de reproducción, tomada de Li et al. (2010) y traducida al español)	42
5.1	Arquitectura de la solución planteada con Docker, Mesos y Marathon	48
5.2	LPT-CRD: Distribución de imágenes y CPUs en base a criterios de tamaño y velocidad, respectivamente	50
5.3	CCRD: Esquema de asignación que permite minimizar el mean flowtime	52
5.4	Especificación de una tarea Marathon correspondiente a nodos lentos (utilizando 60 % de la CPU)	53
5.5	Especificación de una tarea Marathon correspondiente a nodos rápidos (utilizando 100 % de la CPU)	53
5.6	Esquema de funcionamiento de Azure Batch [imagen de dominio público, tomada de Myers et al. (2017) y traducida al español] .	55
5.7	Diagrama de secuencia del programa ejecutado en Azure	56
5.8	Flujo de obtención de acceso a Azure Blob Storage	59
5.9	Arquitectura para procesar archivos de salida	59
5.10	Tablas utilizadas para el almacenamiento de los resultados	60
5.11	Esquema de particionamiento de Azure Tables [imagen de dominio público, tomada de Myers (2017)]	61
5.12	Arquitectura de la solución que permite procesar las imágenes astronómicas en Azure Batch y almacenar los resultados en un formato accesible a los investigadores	61
6.1	Speedup de la implementación en Mesos para diferente cantidad de nodos de ejecución en diferentes conjuntos de imágenes	66
6.2	Speedup obtenido con los algoritmos LPT-CRD y CCRD para diferentes conjuntos de imágenes	68

6.3 Aceleración obtenida con los algoritmos LPT-CRD y CCRD para diferentes conjuntos de imágenes	69
--	----

Listas de tablas

2.1	Configuraciones de las cámaras del WFPC2	16
2.2	Resumen de datos disponibles por cada instrumento	18
6.1	Tiempo total de ejecución de la implementación en Mesos para diferentes conjuntos de imágenes, en minutos	64
6.2	Análisis de la eficiencia computacional de la implementación sobre Mesos para 5 y 10 nodos y diferentes conjuntos de imágenes	64
6.3	Resultados de makespan y análisis de eficiencia para la implementación de los algoritmos propuestos sobre diferentes conjuntos de imágenes	66
6.4	Resultados de flowtime y análisis de eficiencia para los algoritmos LPT-CRD y CCRD sobre diferentes conjuntos de imágenes	67
6.5	Métricas de ejecución de tareas en Azure, por unidad de tiempo	70
6.6	Resultados de makespan y análisis de eficiencia para la ejecución en Azure comparada con CCRD, sobre diferentes conjuntos de imágenes	70
6.7	Tiempo medio de procesamiento y cantidad media de rayos cósmicos detectados por instrumento	71

Tabla de contenidos

Lista de figuras	IX
Lista de tablas	xii
1 Introducción	1
2 Detección de rayos cósmicos con imágenes del HST	10
2.1 La detección de Rayos Cósmicos	10
2.2 El telescopio espacial Hubble y sus instrumentos	12
2.2.1 Historia	12
2.2.2 Instrumentos	13
2.3 El formato FITS	19
2.4 Software de procesamiento astronómico	21
3 Computación de alto desempeño, computación cloud y almacenamiento de datos	25
3.1 Cloud computing	25
3.2 Computación de alto desempeño	29
3.2.1 Modelos de computación de alto desempeño	29
3.2.2 Modelos de computación paralela	29
3.2.3 Computación de alto desempeño y computación paralela en el problema del análisis masivo de imágenes astronómicas	31
3.3 Almacenamiento, replicación y disponibilidad de los datos . . .	32
4 Trabajos Relacionados	36
5 Arquitectura propuesta	44
5.1 Arquitectura utilizando Apache Mesos y Marathon	44

5.1.1	Pipeline de procesamiento de imágenes utilizando Mesos y Marathon	44
5.1.2	Mejoras en el algoritmo de planificación	48
5.2	Arquitectura utilizando Microsoft Azure	54
5.3	Manipulación de los datos	57
6	Análisis Experimental	62
6.1	Métricas	62
6.2	Implementación en Apache Mesos	63
6.3	Microsoft Azure	68
7	Conclusiones y Trabajo Futuro	73
7.1	Conclusiones	73
7.2	Trabajo Futuro	75
	Referencias bibliográficas	78

Capítulo 1

Introducción

Los rayos cósmicos son partículas cargadas de energía cuyo origen puede asociarse a las tormentas solares y a la radiación proveniente de los confines del universo ([Christian, 2012](#)). Su estudio permite comprender la intensidad de fenómenos tales como los vientos y las erupciones solares. Los rayos cósmicos son un problema para los instrumentos científicos utilizados para mediciones astronómicas, en particular para los instrumentos que están expuestos a los rayos cósmicos toda vez que son enviados al espacio exterior. La radiación provocada por los rayos cósmicos reduce la vida útil y afecta el funcionamiento de los instrumentos astronómicos, entre otros problemas ([Hora et al., 2006](#)). Por este motivo, las diversas misiones de investigación deben considerar protecciones especiales para los sensores que se verán expuestos a diferentes tipos de radiación cósmica una vez atravesada la atmósfera terrestre. Sin embargo, el problema de los rayos cósmicos no es exclusivo de la estratosfera. Los instrumentos instalados en nuestro planeta también sufren las consecuencias del bombardeo de rayos cósmicos, y es frecuente que cascadas (o lluvias) de rayos cósmicos alcancen la Tierra e impacten los instrumentos instalados a nivel terrestre.

El planeta Tierra está protegido naturalmente contra los rayos cósmicos a través de su magnetósfera. Aunque la atmósfera es capaz de obstruir el paso de rayos cósmicos de baja energía (low-energy cosmic rays), es la magnetósfera quien bloquea los rayos cósmicos de alta energía (high-energy cosmic rays). La radiación provocada por los rayos cósmicos de alta energía es mortal para la vida en la Tierra. Por este motivo es importante entender cuál es el estado del campo magnético en términos de su fortaleza y de su capacidad de rechazar la

radiación cósmica. Para obtener una estimación de qué tan bien está funcionando este mecanismo natural es necesario cruzar datos obtenidos a nivel del suelo con datos obtenidos por encima de la atmósfera, y también con información relativa a la actividad solar. Es muy difícil establecer la relación entre la actividad solar y el impacto de los rayos cósmicos en la atmósfera sin contar con información adicional a la que proporcionan los detectores instalados a nivel del suelo. Por este motivo es necesario desplegar instrumentos de medición por encima de la atmósfera para tener una idea más cabal de a qué tipo de fenómenos relacionados con los rayos cósmicos se exponen los seres vivos en la Tierra y por consiguiente estimar la fortaleza de la magnetósfera.

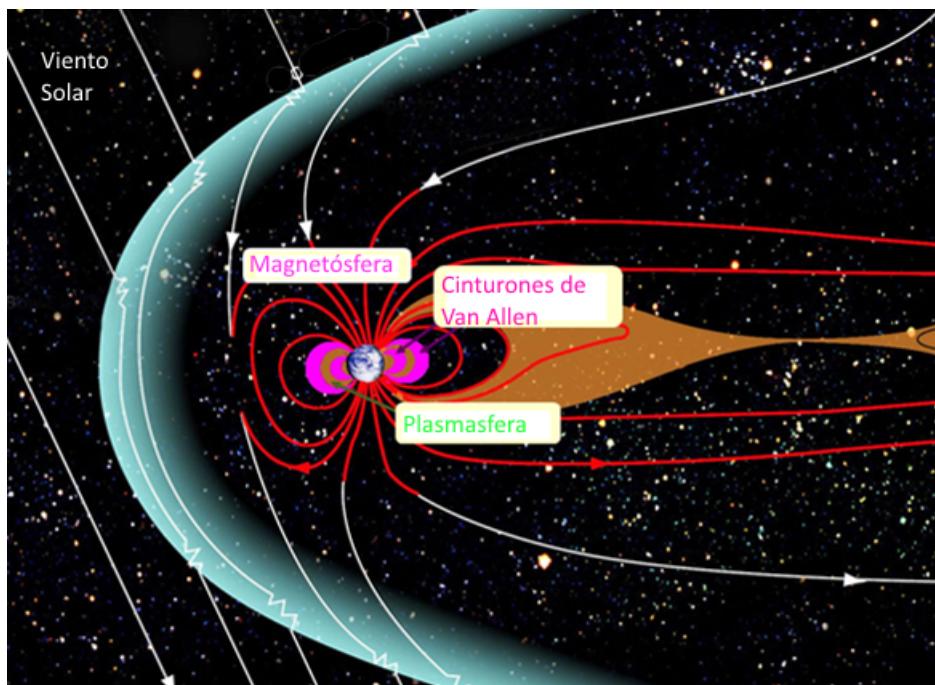


Figura 1.1: Magnetósfera de la Tierra (imagen de dominio público, crédito NASA/Goddard/Aaron Kaase)

Las regiones más importantes del campo magnético para comprender las variaciones de los fenómenos solares se denominan cinturones de Van Allen, en honor a su descubridor. La Figura 1.1 muestra la ubicación de los cinturones de Van Allen en relación a la magnetósfera y al planeta Tierra. Los cinturones de Van Allen son regiones donde los vientos solares y los rayos cósmicos quedan atrapados y conforman un campo energético que combate la radiación en general. En particular, existe el fenómeno que se denomina anomalía del Atlántico

Sur (SAA, por sus siglas en inglés). La Figura 1.2 muestra la ubicación de la SAA con datos tomados por el detector SAAD desplegado a tal fin por la misión ROSAT ([Zimmermann, 1992](#)). El SAA conforma el punto más cercano de la Tierra al cinturón de Van Allen más interior. Todos los satélites que orbitan alrededor de la Tierra pasan por esta región y sus mediciones en esta zona son muy imprecisas y cargadas de ruido debido a la alta radiación atrapada que afecta la sensibilidad de los instrumentos. Entre los países afectados por la SAA se encuentra Uruguay.

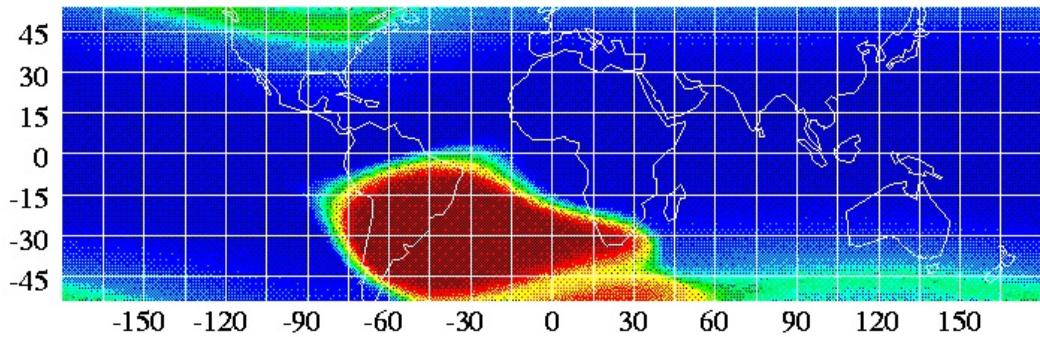


Figura 1.2: Anomalía del Atlántico sur (imagen de dominio público, tomada de [Snowden \(2011\)](#))

Sin embargo, mirada desde otro punto de vista, la problemática de la sensibilidad de los instrumentos de investigación astronómica a la radiación cósmica puede resultar beneficiosa. Los instrumentos que fueron enviados al espacio exterior son capaces de proveer información relativa al impacto de rayos cósmicos de alta energía para la estimación del estado del campo magnético. En particular, un telescopio notable es apropiado para esta tarea: el telescopio espacial Hubble (HST por sus nombre en inglés, Hubble Space Telescope). El HST tiene la característica de poseer varios instrumentos a bordo y de orbitar el planeta sobre las regiones para las que no se dispone de información sobre el impacto de rayos cósmicos. De esta manera, utilizando las capturas de imágenes que se realizan periódicamente por los instrumentos del HST se puede determinar el impacto de los rayos cósmicos en la atmósfera terrestre.

El HST dispone de múltiples instrumentos, a través de los cuales los investigadores han logrado recolectar más de 25 años de datos astronómicos. Los instrumentos del HST son de diferente tipo (cámaras, espectrógrafos, etc.) y como consecuencia recolectan información diferente en cada caso. Si bien la radiación cósmica afecta a todos los instrumentos, las trazas del impacto de

los rayos cósmicos es evidente en las imágenes de las cámaras a bordo del HST. En particular, el contraste más alto entre las trazas de los rayos y el resto de la información capturada por la cámara se da en las imágenes de tipo *dark*. Los darks son el tipo de fotografía que toman las cámaras del HST para calibrar diferentes parámetros de cada uno de los instrumentos, y tienen la particularidad de que se toman con el lente tapado. La imagen de tipo dark es una imagen negra. Sin embargo, los rayos cósmicos atraviesan cualquier superficie, incluso la protección del lente de la cámara que está siendo calibrada, por lo que la imagen de tipo dark preserva la traza de los rayos cósmicos que impactaron en el momento de apertura del obturador. Como consecuencia, este conjunto de imágenes de tipo dark contiene información relevante para el estudio de los rayos cósmicos por fuera de la protección de la atmósfera terrestre. Además, el HST orbita la SAA por lo que los datos obtenidos a través de los darks de sus cámaras cumplen con las características necesarias para contrastar la información de la actividad solar, la radiación cósmica por encima de la atmósfera y la radiación a nivel del suelo terrestre considerando las variaciones del campo magnético en cada región.

En el área de tratamiento de imágenes, la traza dejada por el impacto de los rayos cósmicos se considera ruido, y en general se descarta mediante algoritmos de limpieza de la imagen cruda para obtener como resultado una imagen limpia. Para obtener información del impacto de la radiación cósmica debe aplicarse un paso más: restar la imagen limpia a la imagen original. Con este paso extra se obtiene una imagen que contiene las trazas que evidencian el impacto de los rayos cósmicos en el instrumento que tomó la imagen.

Se propone estudiar el conjunto de imágenes de tipo dark correspondiente a la totalidad del período de actividad del HST para estudiar las trazas de los rayos cósmicos que impactaron en sus instrumentos. El volumen de datos correspondiente a este conjunto de imágenes ronda los 10 TB por lo que es necesario evaluar diferentes alternativas en el tratamiento de la información contenida en los archivos correspondientes. Si se considera el potencial de información disponible a través de los darks del HST, el aporte científico del enfoque planteado es mayúsculo. Se dispone de una fuente de datos enorme con una característica muy importante: se extiende en el tiempo, lo que permite analizar la fluctuación de la densidad de la magnetósfera.

En este trabajo se proponen diferentes opciones para el almacenamiento de los datos que serán analizados, de forma de asegurar la accesibilidad a los

mismos en todo momento por parte de los nodos encargados de realizar el procesamiento de las imágenes. Se presenta la posibilidad del almacenamiento local en un servidor donde corre el proceso de análisis, y se presenta la alternativa en un almacenamiento redundante y elástico en la nube.

Para el procesamiento de las imágenes se presentan tres alternativas. En primera instancia se menciona brevemente una solución basada en MapReduce utilizando Hadoop. Esta aproximación fue derivada a otro equipo de trabajo y la comparación de los resultados de este trabajo con los resultados de su investigación quedan planteados como trabajo futuro. El primer análisis profundo del problema del procesamiento masivo se presenta con el diseño de una arquitectura basada en Apache Mesos ([Hindman et al., 2011](#)) y las herramientas de su ecosistema como Marathon ([Mesosphere Inc., 2016](#)) y Zookeeper ([Hunt et al., 2010](#)). Los resultados son satisfactorios en términos generales pero de todas maneras se proponen dos mejoras puntuales en el algoritmo de asignación de tareas a los nodos de cómputo que permiten obtener conclusiones acerca de la escalabilidad de la arquitectura planteada así como de la fiabilidad de la tecnología. En base a las conclusiones obtenidas en los experimentos con Apache Mesos se realizan algunas modificaciones a la arquitectura y se realizan más ejecuciones en Microsoft Azure. Como los resultados de las pruebas preliminares realizadas sobre la plataforma Azure son satisfactorios en todo sentido (redundancia del almacenamiento y tiempo de procesamiento menor a las pruebas en Mesos) se procede a la elaboración de un plan de ejecución de la totalidad del conjunto de imágenes disponibles sobre esta plataforma.

Finalmente, los resultados obtenidos en el procesamiento de las imágenes quedan disponibles en una estructura de datos diseñada específicamente para la consulta por parte de los investigadores interesados. Se construyen dos tablas para almacenar información puntual sobre cada imagen procesada y sobre cada rayo cósmico que impactó en cada imagen. El diseño de las tablas que se presenta contempla que los usuarios puedan consultar los resultados por instrumento, por imagen, por rango de fechas y por otros atributos.

La mayor contribución de esta tesis es el conjunto de datos que resultan del procesamiento de las imágenes del HST. Estos datos contribuyen al estudio de la magnetósfera en diferentes períodos de tiempo y pueden ser utilizados en investigaciones en el área de la geofísica y la astronomía. Además, la arquitectura desarrollada puede ser reutilizada en otros flujos de procesamiento de imágenes astronómicas que tengan formato y características similares a las

utilizadas en esta trabajo.

El contenido de esta tesis está estructurado del modo que se describe a continuación. El próximo capítulo introduce la definición de rayos cósmicos y detalla cómo se obtienen y manipulan a partir de los instrumentos del HST. A continuación, el Capítulo 3 presenta un resumen de prácticas de Cloud Computing y Computación de Alta Performance (HPC, del inglés High Performance Computing) que serán utilizadas en el diseño de la arquitectura que permite resolver el problema del análisis del conjunto de imágenes. El Capítulo 4 presenta una lista de trabajos relacionados con problemáticas similares en algún punto a esta tesis y las soluciones que aportaron dichos trabajos en cada caso. El Capítulo 5 detalla la evaluación de tecnologías como Hadoop, Spark, Mesos y Marathon aplicadas al análisis masivo de las imágenes. El Capítulo 6 presenta los resultados del procesamiento de las imágenes, en términos de tiempos de ejecución y evaluación de aspectos puntuales de cada tecnología utilizada. Finalmente, el Capítulo 7 presenta las conclusiones y las diferentes líneas de trabajo futuro.

Publicaciones a partir de este trabajo

A continuación se presenta un resumen de los trabajos publicados durante la elaboración de esta tesis, con su abstract original y una breve descripción en español.

- G. Schnyder y S. Nesmachnow. Improving the performance of cosmic ray detection using Apache Mesos. En *International Supercomputing Conference in México*, Ciudad de México, México, 2015.

This article presents a parallel approach for image analysis and processing using Image Reduction and Analysis Facility (IRAF) and Docker over Apache Mesos. IRAF is an image analysis software developed by National Optical Astronomy Observatories focused on scientific research. There is a set of images corresponding to several years of investigation that evidence the presence of cosmic rays hitting measuring instruments. Through mathematical analysis provided by IRAF it is possible to determine how strong are cosmic rays, a fact that needs to be considered when designing space instruments. A task is defined as an instance of IRAF

package executing cosmic ray detection over a single image. On this article every task is executed by a separate node deployed on a Mesos managed environment using Docker containers (lightweight virtual machines). These containers allow to optimize the physically available resources usage. Results show that, through a proper distribution and assignment of resources, a lineal improvement is achieved. This is compared to sequential execution on a single node so the improvement is proportional to the number of nodes used for processing.

En este artículo se describe una arquitectura posible para el análisis de las imágenes dark del HST utilizando Apache Mesos. El software que ejecuta el análisis utiliza la biblioteca IRAF y es distribuido en contenedores Docker. Se reportan las mejoras de performance en base al aumento de la cantidad de nodos.

- G. Schnyder, S. Nesmachnow, G. Tancredi y A. Tchernykh. Scheduling algorithms for distributed cosmic ray detection using Apache Mesos. En *CARLA: Latin American High Performance Computing Conference*, Ciudad de México, México, 2016.

This article presents two scheduling algorithms applied to the processing of astronomical images to detect cosmic rays on distributed memory high performance computing systems. We extend our previous article that proposed a parallel approach to improve processing times on image analysis using the Image Reduction and Analysis Facility IRAF software and the Docker project over Apache Mesos. By default, Mesos introduces a simple list scheduling algorithm where the first available task is assigned to the first available processor. On this paper we propose two alternatives for reordering the tasks allocation in order to improve the computational efficiency. The main results show that its possible to reduce the makespan getting a speedup = 4.31 by adjusting how jobs are assigned and using uniform processors.

En este artículo se presentan dos algoritmos de asignación de tareas para el procesamiento de imágenes astronómicas utilizando la plataforma Apache Mesos. El software que ejecuta el análisis utiliza la biblioteca IRAF y es distribuido en contenedores Docker. Se reportan y analizan las mejoras de performance al utilizar los algoritmos de asignación construidos.

- G. Cromwell, S. Deustua, G. Gonzalez Sprinberg, S. Nesmachnow, G. Schnyder y G. Tancredi. Geophysics using Hubble Space Telescope. Hubble Space Telescope Cycle 24 AR Proposal.

We exploit Hubble Space Telescope (HST) as a cosmic ray detector to probe Earth's external magnetic field through analysis of the cosmic ray flux on HST instruments. We propose to analyze 100 000+ dark images obtained during 26 years of HST operation to calculate the flux of cosmic rays at an altitude of 500 km above the surface and estimate variations in the external magnetic field, thereby complementing geophysical observatory measurements. Our analysis will combine HST results with measurements of solar activity, cosmic ray flux on Earth's surface, and geomagnetic data to tease out external field variations. The addition of 26 years of HST data, including 10 years of measurements predating geomagnetic satellites, will provide invaluable, high-resolution observations of the geomagnetic field.

Este reporte técnico presenta la solicitud de apoyo económico enviada con motivo del ciclo de propuestas número 24 del HST. En esta propuesta se mencionan los conjuntos de imágenes que serán analizados, se realiza la justificación científica de la importancia del análisis del impacto de los rayos cósmicos en los instrumentos HST en comparación con datos obtenidos por sensores a nivel del suelo terrestre y se detallan las etapas subsecuentes de la investigación en el área de la geofísica con foco en el campo magnético de la Tierra.

- S. Nesmachnow, G. Tancredi, G. Schnyder, S. Roland y S. Deustua. Cloud Computing and HPC applied to cosmic ray detection in spatial telescope images. Reporte Técnico, Microsoft Azure Research Award.

Solicitud de recursos computacionales para la ejecución del software creado en esta tesis con el fin de analizar la totalidad de las imágenes de tipo dark del HST. En esta propuesta se presenta una breve justificación científica de la importancia de los resultados que se quieren obtener, y se detallan los recursos computacionales estimados para el almacenamiento y el procesamiento de las imágenes.

- G. Schnyder, S. Nesmachnow y G. Tancredi. Distributed cosmic ray detection using cloud computing. En *CARLA: Latin American High Per-*

formance Computing Conference, Buenos Aires, Argentina / Colonia, Uruguay, 2017.

This article presents a distributed computing approach for detecting cosmic rays from images taken by the Hubble Space Telescope (HST). A cloud computing implementation is developed to improve the overall processing time for the available images dataset (15 TB), containing dark images from several HST instruments. A specific architecture is presented where the images are stored in a replicated and highly available storage system. Image processing is performed on virtual machines from the Azure Batch framework using a developed Python application. The experimental evaluation shows that the architecture accomplished the purpose of processing the complete dataset based on scaling computing resources in terms of processing nodes. Speedup improved in a factor of 6.57× over a previous implementation using Apache Mesos. The overall computation took 10 days to complete and results are stored on a non-relational database available to astronomers and researchers.

En este artículo se presenta un enfoque de cloud computing para el procesamiento de imágenes astronómicas utilizando el proveedor de servicios Microsoft Azure. El software que ejecuta el análisis está desarrollado en Python y es distribuido a los nodos de procesamiento utilizando Azure Batch. Las imágenes son almacenadas en Azure Blob y los resultados en Azure Table. Se reportan y analizan las mejoras de performance al utilizar la plataforma cloud como alternativa a las ejecuciones realizadas en servidores locales.

Capítulo 2

Detección de rayos cósmicos con imágenes del HST

Este capítulo presenta la utilización de los instrumentos del HST para la detección de rayos cósmicos mediante el análisis de imágenes astronómicas con software específico.

2.1. La detección de Rayos Cósmicos

Los rayos cósmicos son partículas energéticas que viajan a través del universo. Existen tres tipos de rayos cósmicos: los galácticos, los anómalos y los solares ([Mewaldt *et al.*, 1994](#)). La radiación cósmica es un tipo de radiación que afecta el funcionamiento de dispositivos electrónicos de una forma comprobable y medible. En particular, los instrumentos del HST están expuestos a este fenómeno y por lo tanto son una fuente muy rica de detección de rayos cósmicos de baja energía, dado que no están bajo la protección de una atmósfera como la terrestre.

La detección de rayos cósmicos se puede realizar de varias maneras, entre las que se encuentran la instalación de dispositivos especiales a nivel terrestre o el envío de instrumentos al espacio. En general no es un problema el acceso a registros de impacto de rayos cósmicos debido a que la radiación cósmica es capaz de atravesar todo tipo de material e impactar sobre los sensores de los dispositivos instalados para el registro de los rayos cósmicos. El problema que surge en la captación de radiación cósmica es que es necesario estudiar las variaciones causadas por campos magnéticos como la magnetósfera. Al atra-

vesar la magnetósfera los rayos cósmicos pierden energía, luego atraviesan la atmósfera terrestre y finalmente alcanzan el suelo terrestre en forma de cascada. El estudio de la radiación que llega al suelo terrestre aporta información valiosa desde el punto de vista científico, pero no es posible determinar con suficiente certeza el estado de la magnetósfera. Para poder determinar qué tan fuerte se encuentra el campo magnético de la tierra en términos de rechaza de la radiación cósmica es necesario estudiar la trayectoria de los rayos cósmicos en una etapa anterior al impacto con la atmósfera. El HST, a través de los instrumentos que lleva a bordo, permite obtener información relativa a la fuerza que poseen los rayos cósmicos luego de atravesar la magnetósfera y antes de atravesar la atmósfera.

En la Figura 2.1 se aprecia el impacto de partículas energéticas en forma de rayos cósmicos sobre el lente de un instrumento del HST (en particular, la Advanced Camera for Surveys o ACS). Los píxeles afectados en la imagen revelan información fundamental sobre los rayos cósmicos que impactaron. Mediante el uso de software específico, como el que se describe en la Sección 2.4, y utilizando el formato de imágenes FITS (del inglés, Flexible Image Transport System), se pueden extraer las características de cada rayo que impactó en el instrumento. Con la información recogida en la imagen se puede determinar el punto geográfico y la ubicación del impacto del rayo cósmico, y construir un mapa de incidencia de rayos cósmicos a nivel terrestre. Para la construcción de este mapa se deben ubicar las trazas de los impactos con respecto a la atmósfera terrestre, en particular especificando longitud, latitud y altitud. Para determinar la ubicación de las trazas con respecto a las coordenadas terrestres se utiliza una conversión predeterminada que se aplica a las coordenadas celestes correspondientes al posicionamiento del instrumento al momento de tomar la imagen y que determina los valores terrestres. En particular, las coordenadas que corresponden a los ejes cartesianos X, Y y Z vienen por defecto especificadas en el sistema de coordenadas inerciales geocéntricas J2000.0 ([Feissel y Mignard, 1998](#)). J2000.0 es un sistema de coordenadas de mano derecha con centro en la tierra, con el eje X apuntando hacia el equinoccio vernal para el año 2000, con el eje Z apuntando hacia el polo norte celeste para el año 2000 y el eje Y ortogonal a ambos. Como resultado, una vez procesadas todas las imágenes se obtienen los datos necesarios para realizar un mapa a nivel global del impacto de rayos cósmicos y por consiguiente una forma de entender qué tan debilitado (o fortalecido) está el campo magnético en cada región.



Figura 2.1: Imagen dark con impactos de rayos cósmicos (sección de j6mf16lhq-raw.fits)

2.2. El telescopio espacial Hubble y sus instrumentos

Esta sección presenta una descripción del Telescopio Espacial Hubble y sus instrumentos.

2.2.1. Historia

El HST fue lanzado el 24 de abril de 1990. A lo largo de su período operativo ha realizado más de un millón de observaciones, y es el telescopio especial mediante el cual se han hecho algunos de los descubrimientos más significativos en astronomía. Con el fin de mantenerlo en actividad, ha sido objeto de múltiples misiones de servicio para reparar sus instrumentos o actualizarlos ([Space Telescope Science Institute, 2017b](#)). En la Figura 2.2 se detalla el período operativo de cada instrumento, donde se observa que algunos instrumentos son muy recientes (en particular la Wide Field Camera 3 y el Cosmic Origins Spectrograph) y otros estuvieron en el periodo inicial del HST pero rápidamente fueron sustituidos. Por ejemplo, el caso más notorio es el de la

WFPC (Wide Field and Planetary Camera) que era el instrumento designado para tomar las imágenes más significativas de la investigación espacial pero por una falla en su construcción tuvo que ser sustituido por la WFPC2 (Wide Field and Planetary Camera 2).

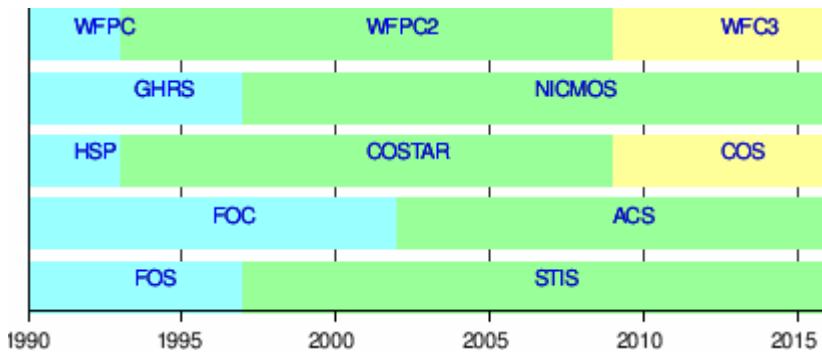


Figura 2.2: Período operativo de cada instrumento del HST (imagen de dominio público, tomada de Wikipedia)

2.2.2. Instrumentos

Los instrumentos del HST conforman la parte esencial del telescopio. En el HST existen instrumentos de tipo cámara, algunos espectrógrafos y diversos tipos de sensores, pero en particular en este trabajo se utilizan las imágenes de aquellos instrumentos que por sus características son capaces de registrar el impacto de rayos cósmicos. Los instrumentos del HST funcionan de forma síncrona, de manera que las observaciones de los diferentes instrumentos suceden al mismo tiempo y sobre el mismo haz de luz reflejado por el espejo principal. Cada instrumento analiza un rango del espectro de la luz específico. En las Figuras 2.3a y 2.3b se muestra cómo accede la luz al interior del HST y se ve reflejada en una serie de espejos colocados con el fin de que todos los instrumentos tengan acceso a ella. La disposición interna de los instrumentos puede apreciarse en la Figura 2.4.

A continuación se presenta una lista de los instrumentos que capturaron las imágenes analizadas en este trabajo, con una breve descripción técnica e histórica en cada caso.

ACS

ACS (del inglés, Advanced Camera for Surveys) es un instrumento compuesto por tres cámaras de diferente tipo: wide-field camera (WFC), high re-

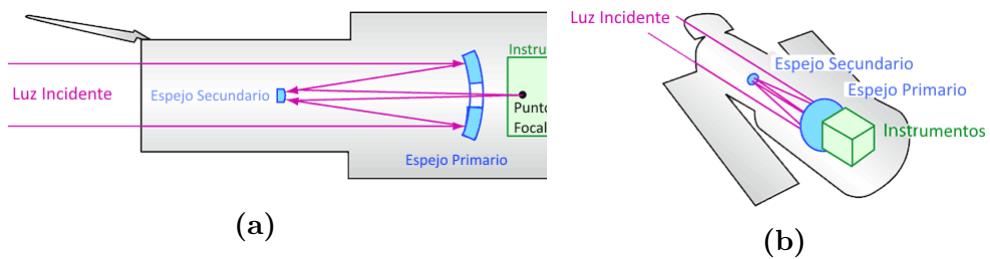


Figura 2.3: Captura de la luz por parte del HST (imagen de dominio público, tomada de [Space Telescope Science Institute, 2017a](#))

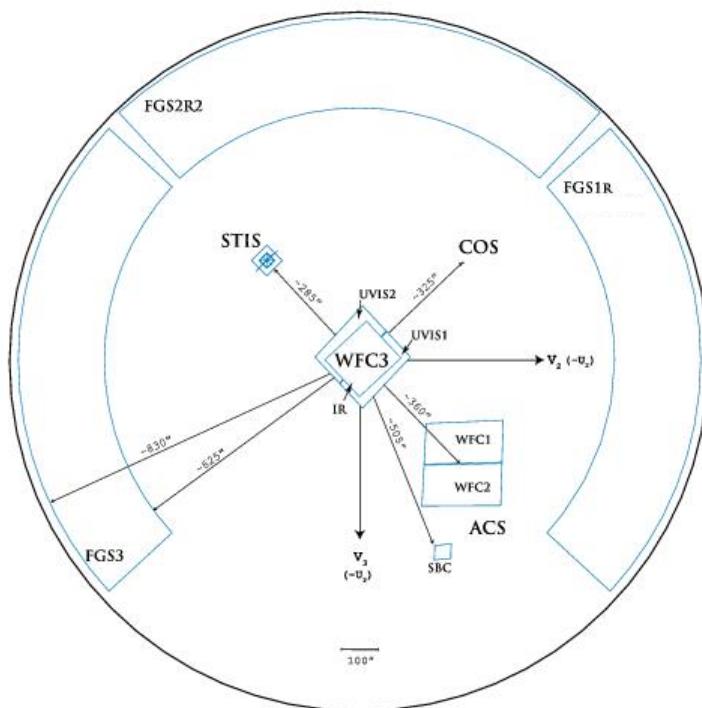


Figura 2.4: Disposición física de los instrumentos del HST (imagen de dominio público, tomada de [Riley y Biretta, 2017](#))

solution camera (HRC), y solar blind channel (SBC). A partir de una falla ocurrida en enero de 2007, la WFC y la HRC dejaron de funcionar por un período aproximado de dos años. Durante la cuarta misión de mantenimiento del HST se pudo recuperar la WFC pero no la HRC, por lo que no está disponible para su uso a partir de entonces. Las imágenes de la ACS tienen un tamaño de 2048×4096 píxeles. Los datos de la exposición de las cámaras están en el archivo con extensión `flt` (con el overscan ya eliminado) y los datos de posicionamiento del instrumento están en el archivo con extensión `spt`.

COS

COS (del inglés, Cosmic Origins Spectrograph) es un espectrógrafo ultravioleta que fue instalado en 2009 como reemplazo del dispositivo de corrección denominado COSTAR, y con el objetivo de complementar a otros espectrógrafos existentes. Tiene dos canales para la captación de las ondas, el FUV (far ultraviolet) y el NUV (near ultraviolet) y está pensado para observar puntos luminosos como estrellas y quásars. Las imágenes de COS tienen un tamaño de 16384×1024 píxeles y vienen en dos archivos separados con extensiones `flt_a` y `flt_b` en caso de FUV y un solo archivo `flt` en caso de NUV. Los datos de posicionamiento del instrumento están en el archivo con extensión `spt`.

NICMOS

NICMOS (del inglés, Near Infrared Camera and Multi-Object Spectrometer) es un instrumento con capacidad para captura de imágenes en infrarrojo y observación espectroscópica de objetos astronómicos. Tuvo dos períodos de funcionamiento: el primero desde 1997 hasta 1999, y el segundo desde 2002 hasta 2008. Quedó en desuso cuando se instaló la WFC3, pero durante su período de funcionamiento era el instrumento encargado del espectro cercano al infrarrojo. Las imágenes del NICMOS tienen un tamaño de 256×256 píxeles y vienen consolidadas en una sola imagen generada mediante el pipeline interno de STScI, en particular por la utilidad `calnicb`.

STIS

STIS (del inglés, Space Telescope Imaging Spectrograph) es un espectrógrafo que dispone también de una cámara. Fue instalado en 1997 y funcionó sin problemas hasta el año 2004 donde interrumpió su funcionamiento debido a una falla eléctrica. Fue reparado en 2009 y continúa en uso hasta la fecha. Está especializado en el espectro ultravioleta de la luz y fue responsable, entre otras cosas, del análisis de la primera atmósfera en un planeta extrasolar ([Charbonneau et al., 2002](#)). Es el instrumento utilizado en la búsqueda de agujeros negros debido a su espectro (por ejemplo superior al de COS) y la propiedad de corrimiento al rojo o al azul que presenta la luz de las estrellas y nubes de gas cuando se acercan o se alejan del punto de observación. Posee tres detectores que generan tres imágenes de 1024×1024 píxeles. Los datos de observación vienen en el archivo con extensión `flt` (con las imágenes ya agregadas en una

sola imagen final sin overscan) mientras que los datos de posicionamiento del instrumento están en el archivo con extensión spt.

WFPC2

WFPC2 (del inglés, Wide Field Planetary Camera 2) era un instrumento para captura de imágenes que estuvo instalado en el HST en un período de unos 26 años. Suplantó a la WF/PC-1 y fue suplantada a su vez por la WFC3. Es el instrumento que tomó las fotos más famosas del HST, como por ejemplo The Pillars of Creation (Figura 2.6a) y Hubble Deep Field (Figura 2.6b). Consistía de tres sensores wide-field (WFC, del inglés Wide Field Camera) en forma de L y una cámara de alta resolución (PC, del inglés Planetary Camera). Una imagen típica de la WFPC2 presenta cuatro componentes de 800×800 píxeles, debido a las 3 WFC y a la PC. Esta última suele aparecer más pequeña en las imágenes debido a su menor campo visual como se detalla en la Tabla 2.1.

Cámara	Píxeles	Campo Visible	Escala
PC	800×800	$36'' \times 36''$	$0.0455''$ por pixel
WF2, 3, 4	800×800	$80'' \times 80''$	$0.0996''$ por pixel

Tabla 2.1: Configuraciones de las cámaras del WFPC2

Con respecto a los archivos disponibles para esta cámara, las extensiones c0m corresponden a la imagen ya reconstruida y calibrada (es decir, con todos las capturas de cada instrumento incluidas y el overscan eliminado) por lo que es la entrada ideal para el algoritmo implementado. Este archivo contiene además datos específicos de cada captura en forma de extensión del formato FITS. La información de posición y parámetros de observación se encuentra en el archivo con extensión shm.

WFC3

WFC3 (del inglés, Wide Field Camera 3) es un instrumento con capacidad para captura de imágenes en infrarrojo (IR) y ultravioletas (UVIS). Fue instalado en 2009 con el objetivo de reemplazar a la WFPC2. Mediante un selector de canal se le indica en los parámetros de observación qué detector se quiere utilizar (IR o UVIS). Las imágenes de la WFC3 tienen un tamaño de 2051×4096 píxeles. Los datos de la exposición de las cámaras están en el

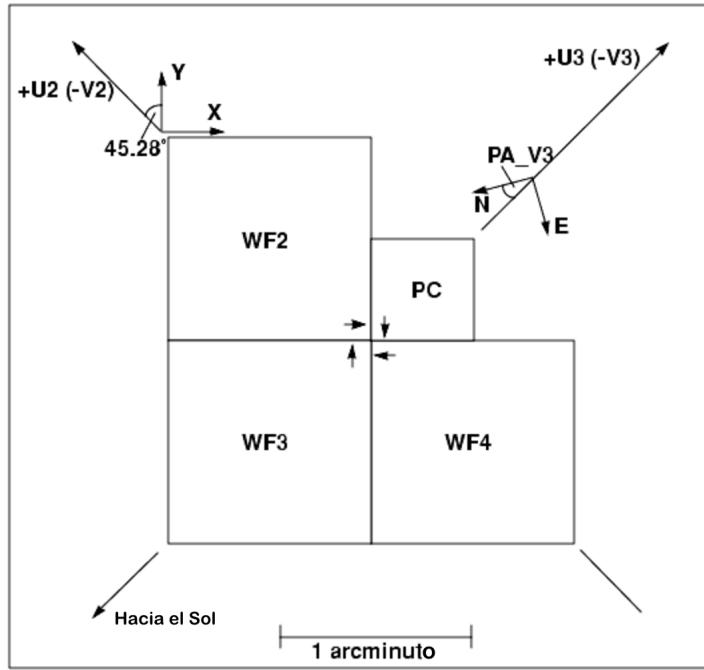


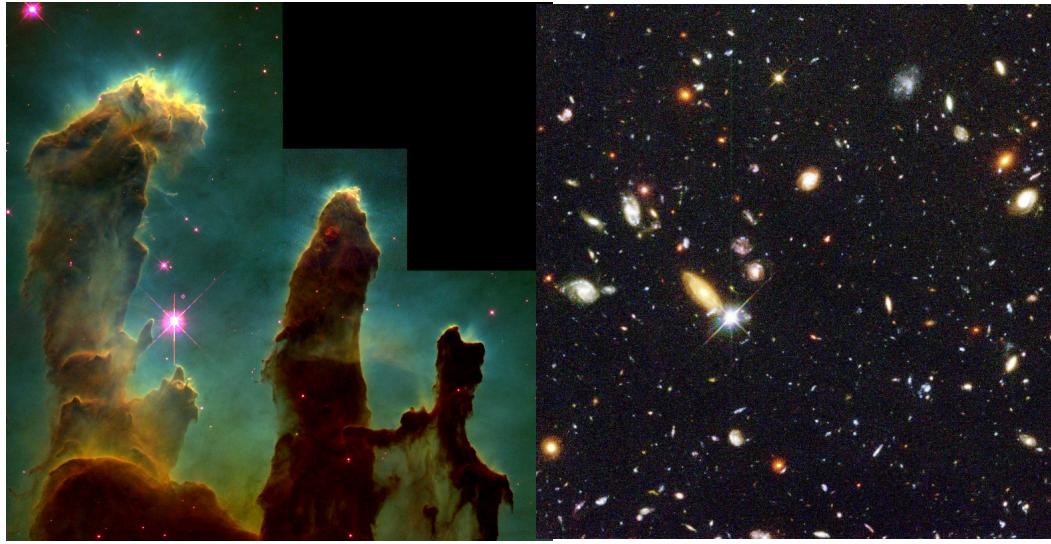
Figura 2.5: Estructura de la WFCP2 (imagen de dominio público, tomada de [McMaster y Biretta, 2008](#))

archivo con extensión flt (con el overscan ya eliminado) y los datos de posicionamiento del instrumento están en el archivo con extensión spt.

Otros instrumentos del HST

Durante su largo período operativo el HST ha contado con otros instrumentos cuyas características no serán detalladas pero que se mencionan brevemente a continuación:

- FGS, Fine Guidance Sensors, sensores para posicionar el HST
- FOC, Faint Object Camera, operativa desde 1990 hasta 2002
- FOS, Faint Object Spectrograph, operativo desde 1990 hasta 1997
- GHRS, Goddard High Resolution Spectrograph, operativo desde 1990 hasta 1997
- HSP, High Speed Photometer, operativo desde 1990 hasta 1993
- WF/PC-1, Wide Field Planetary Camera 1, operativa desde 1990 hasta 1993



(a) Pillars of Creation

(b) Hubble Deep Field

Figura 2.6: Imágenes obtenidas por la WFPC2 [imagen de dominio público, crédito: NASA/ESA/STScI/Arizona State University]

Resumen de instrumentos del HST

La Tabla 2.2 resume los datos disponibles para cada instrumento, a saber: en la columna ".archivos" se listan los extensiones correspondientes a los datos crudos y a los datos de posicionamiento del instrumento; en la columna "tamaño de imagen" se especifica el tamaño en píxeles (y en formato ancho×altura) de cada imagen resultante; y en la columna "período disponible" se muestra qué periodo de datos fue disponibilizado en el acceso a los archivos (y por lo tanto representa un subconjunto del período total de funcionamiento de cada instrumento).

instrumento	archivos datos	archivos posición	tamaño de imagen (en píxeles)	período disponible
ACS	flt	spt	2048×4096	2002-2014
COS	flt_a, flt_b	spt	16384×1024	2009-2016
NICMOS	raw, mos	spt	256×256	1997-2009
STIS	flt	spt	1024×1024	1997-2016
WFC3	flt	spt	2051×4096	2009-2014
WFPC2	c0m	shm	800×800	1993-2009

Tabla 2.2: Resumen de datos disponibles por cada instrumento

2.3. El formato FITS

El formato FITS es un formato universal definido para el intercambio de imágenes astronómicas ([Pence *et al.*, 2010](#)). Cada imagen FITS está conformada por uno o más HDU (del inglés Header Data Unit), donde cada HDU especifica un tipo especial de información. La primera parte de un HDU declara entradas de tipo <clave, valor> que determinan la estructura del contenido binario que sigue a continuación en el archivo FITS. Estas entradas contienen información diversa, desde la configuración del instrumento utilizado para la captura de la imagen hasta datos del investigador que solicitó la captura . Una imagen FITS con un solo HDU se denomina “Basic FITS” y constituye la mínima expresión del formato. A partir de esta forma básica se pueden agregar extensiones para enriquecer la información relacionada a la imagen capturada. La Figura 2.7 presenta la arquitectura interna de una imagen FITS, donde se aprecia el header primario seguido de los datos y finalmente un conjunto variable de extensiones.

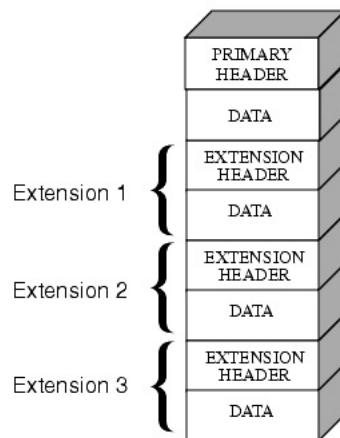


Figura 2.7: Estructura de imagen FITS con extensiones (imagen de dominio público, tomada de [Smith *et al.*, 2011](#))

El formato FITS establece una forma genérica de comunicar datos astronómicos, donde cada instrumento establece cómo va a utilizar las diferentes variantes del formato y qué tipo de headers y extensiones se incluyen como resultado de la observación. Cada observación astronómica tiene como resultado un conjunto de archivos de diferente tipo, y este conjunto de archivos varía de acuerdo al instrumento que realizó la observación. Sin embargo, existen

algunos tipos de archivos que son utilizados por todos los instrumentos para preservar información referente al posicionamiento espacial del instrumento, al tipo de observación y a las manipulaciones que sufrió el dataset cuando fue procesado con el software interno del STScI: los tipos SPT y TRL. Los archivos SPT (del inglés, Support, Planning and Telemetry information) contienen la información del investigador que solicitó la observación original y la propuesta de observación (fechas, telemetría, ubicación, etc.). Los archivos TRL (del inglés, Trailer file) contienen información relativa al tipo de procesamiento que se le realizó al dataset en cuestión y funcionan como salida del sistema cuando se están ejecutando los algoritmos internos de ajuste y análisis.

Una particularidad presente en las imágenes que se obtienen con los instrumentos del HST es el concepto de *overscan*. El *overscan* en una imagen astronómica consiste en información agregada que tiene como fin corregir la desviación que pudo tener el instrumento al momento de tomar la imagen. Durante la realización de la observación programada, el software del instrumento agrega a la imagen columnas y filas que pueden no representar una captura real, pero que sirven para normalizar la observación cuando se procesa la imagen en los servidores del STScI. Este proceso de ajuste posterior se realiza con el software interno de STScI cuando los investigadores solicitan determinado conjunto de imágenes u observaciones, por lo cual en cada dataset completo de una observación se dispone de la imagen cruda y de la imagen corregida. En particular, en este trabajo no se necesitan los píxeles de overscan y por una cuestión de simplicidad se utilizan las imágenes ya corregidas.

En la Figura 2.8 se muestra un diagrama de cómo queda conformada una imagen del instrumento WFC3 que incluye overscan. Cada CCD (del inglés, Charge-coupled device) está compuesto de dos amplificadores, donde cada amplificador observa la región del universo que le es especificada en los parámetros científicos de entrada. La captura tiene un tamaño de 2051×2048 píxeles en cada caso (lo que conforma el tamaño final de 2051×4096 píxeles) pero a estas capturas se le agregan dos tipos de overscan: el vertical y el horizontal. El overscan vertical a su vez tiene dos orígenes: el overscan vertical físico es un área del instrumento (que conforma una franja de 2051×25 píxeles) específicamente diseñada con este propósito, mientras que el overscan vertical virtual es un cálculo de redundancia en base a la observación real. Los overscans horizontales son calculados en el momento de la observación y agregan redundancia que permite luego deshacer la desviación propia de este tipo de procedimientos.

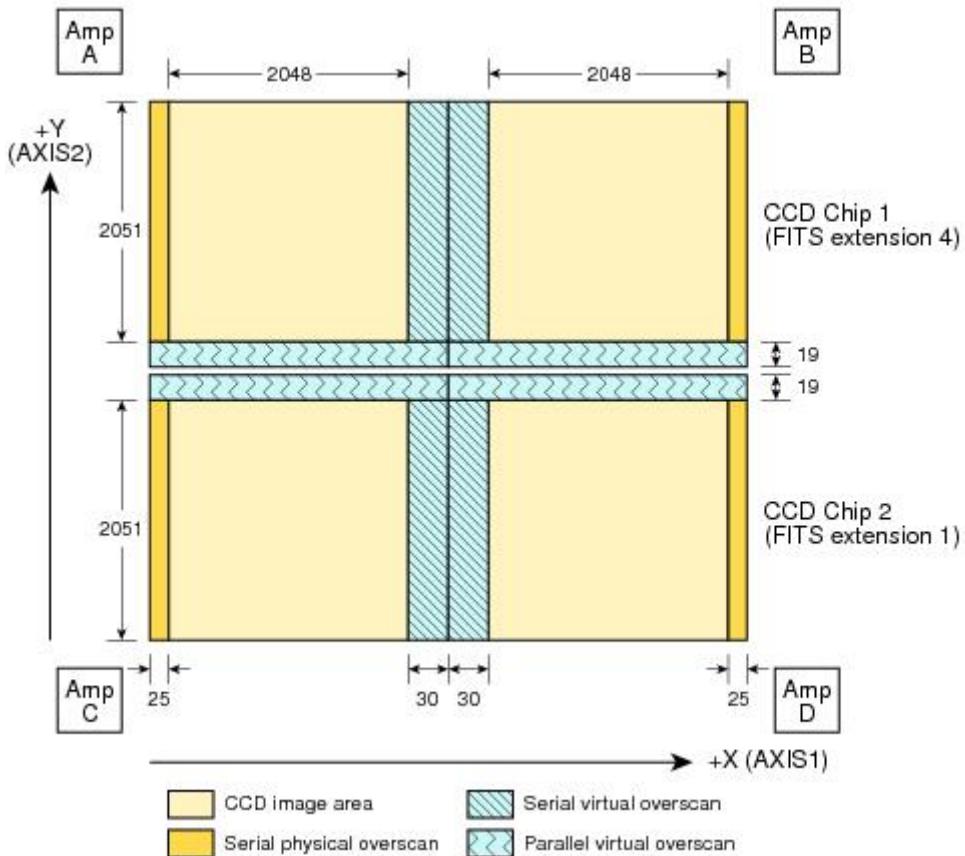


Figura 2.8: Formato de imagen de UVIS con todos los CCD incluidos (imagen de dominio público, tomada de [Dressel, 2017](#))

2.4. Sofware de procesamiento astronómico

El procesamiento de las imágenes con formato FITS requiere la utilización de software específico como IRAF y Astropy. IRAF es un software de propósito general diseñado para procesar información de tipo astronómico. Fue construido, y es mantenido, por un grupo de desarrolladores de NOAO (National Optical Astronomic Observatory). A grandes rasgos, IRAF es un conjunto de herramientas para procesar imágenes y conjuntos de datos astronómicos en general. IRAF permite construir scripts y archivos ejecutables utilizando el entorno de desarrollo y ejecución denominado IRAF Common Language (IRAF CL). La prueba de concepto de la que parte este trabajo está basada en IRAF CL, que interactúa con el sistema de archivos de Linux y un pequeño conjunto de imágenes, con el objetivo de obtener información relevante acerca de los rayos cósmicos en dicho conjunto de imágenes. Como se muestra en el Algoritmo 1, primero se buscan las imágenes en el directorio actual, luego se aplican

filtros de descarte (para asegurarse que la imagen sea de tipo dark entre otras cosas) y finalmente se utiliza la tarea xzap del paquete dimsum para limpiar los rayos cósmicos. Una vez obtenida la imagen limpia, se resta de la original y se obtiene una imagen compuesta solamente de rayos cósmicos.

Algoritmo 1: Obtención de rayos cósmicos usando xzap

```

begin
  foreach imagen en el directorio actual do
    if imagen IS DARK and CHINJECT != NONE and
      FLASHCUR != LOW then
        headers := getHeaders(imagen);
        cleanImage := dimsum.xzap(imagen, ... );
        crs := imagen - cleanImage;
    end
  end

```

En el Código 2.1 se presenta un ejemplo de configuración de la tarea xzap para el análisis de imágenes utilizando IRAF. Los parámetros indican diferentes aspectos a considerar del algoritmo, como valor de mediana de los filtros, cantidad de píxeles a utilizar como buffer alrededor de la imagen y de los rayos cósmicos, la cantidad máxima de iteraciones a ejecutar, etc.

```

xzap.zmin = 2000.      #
Minimum data value for fmedian filter
xzap.zmax = 4000.      #
Maximum data value for fmedian filter
xzap.zboxsz = 10       # Box size for fmedian filter
xzap.skyfiltsize = 15  #
Median filter size for local sky evaluation
xzap.subsample = 0      #
Block averaging factor before median filtering
xzap.ngrowobj = 3       #
Number of pixels to flag as buffer around objects
xzap.nrings = 0         #
Number of pixels to flag around CRs
xzap.nsigneg = 0.       #
Number of sky sigma for negative zapping
xzap.nsigobj = 1.5     #
Number of sky sigma for object identification

```

```

xzap.nsigrej = 2.0      #
The n-sigma sky rejection parameter
xzap.maxiter = 20       # The maximum number of iterations
xzap.statsec = ""        #
Image section to use for computing sky sigma
xzap.nsigzap = 1.        #
Zapping threshold in number of sky sigma

```

Código 2.1: Ejemplo de configuración de tarea xzap

Astropy se define como un paquete de funcionalidades y herramientas comunes para investigación astronómica y astrofísica en Python. En este trabajo se utiliza específicamente para la manipulación de archivos FITS ([The Astropy Collaboration *et al.*, 2013](#)). A través de la herramienta IRAF CL se puede trabajar con archivos FITS en el contexto de IRAF pero en los últimos años el mundo científico ha comenzado a utilizar lenguajes de programación de propósito general con más frecuencia. Como consecuencia de la adopción de los lenguajes de propósito general se han desarrollado paquetes especiales para resolver cuestiones comunes en la investigación científica y en este contexto se enmarca la creación de Astropy. El paquete Astropy contiene varias utilidades para procesamiento de archivos con formatos científicos, entre las que se destaca `astropy.io.fits`. Esta utilidad permite leer los archivos que cumplen el formato FITS así como cargar las extensiones y los headers de los archivos como tipos de datos nativos de Python. Utilizar la representación de datos propia de `astropy` facilita la construcción de abstracciones para la manipulación de los archivos correspondientes a las imágenes de cada instrumento del HST, ya que se dispone de clases que permiten mapear los HDU, extensiones y headers en forma concreta. Además, el paquete `astropy.io.fits` contiene utilidades que facilitan la implementación de tareas recurrentes como parsing de valores y remoción de espacios (y otro tipo de caracteres como separadores y comentarios) que aparecen con frecuencia en los archivos FITS. En relación al posicionamiento espacial de los instrumentos del HST, el paquete Astropy contiene utilidades para el cálculo de coordenadas en diferentes sistemas de referencia a través del paquete `astropy.utils.iers`. La IERS (del inglés, The International Earth Rotation and Reference Systems Service) es responsable de la medición de la orientación terrestre en función del tiempo, además de predecir dicha orientación en el futuro cercano a través de técnicas de extrapolación. La IERS hace públicos los datos de la orientación terrestre mediante los deno-

minados Boletín A y Boletín B: el Boletín B es una tabla con las observaciones de la orientación terrestre de las últimas décadas hasta el tiempo presente (Astropy utiliza esta tabla para calcular UT1-UTC); el Boletín A acompaña las observaciones contenidas en el boletín B realizando además extrapolaciones en el pasado, en períodos no alcanzados por este último. Astropy detecta automáticamente qué boletín hay que utilizar, y de ser necesario actualiza los valores sincronizando con su servidor ([Astroplan, 2017](#)).

Capítulo 3

Computación de alto desempeño, computación cloud y almacenamiento de datos

Este capítulo introduce las metodologías utilizadas en la tesis para mejorar el desempeño computacional de los algoritmos de procesamiento de imágenes astronómicas y optimizar las estructuras para almacenamiento de datos que se utilizan para el análisis masivo de imágenes. El capítulo presenta las características generales del modelo de computación cloud (*cloud computing*), describiendo sus ventajas y desventajas desde el punto de vista del proyecto desarrollado en esta tesis; a continuación se describen los principios de la Computación de alto desempeño (High Performance Computing, HPC) y las principales ideas a aplicar para el manejo de las imágenes desde el punto de vista del software implementado; finalmente se analizan los principales conceptos sobre almacenamiento y manejo de datos a gran escala en sistemas distribuidos.

3.1. Cloud computing

[Grossman \(2009\)](#) definió *cloud* (o nube) como un conjunto de computadores que proveen recursos a demanda con las características de escala y confiabilidad que otorga un datacenter, en términos de alta disponibilidad y capacidad de procesamiento. Para alcanzar los niveles de confiabilidad de un datacenter se utiliza el modelo de *virtualización* de recursos computacionales ([Intel, 2013](#)).

La práctica de virtualización permite abstraer estos recursos de cómputo, en conjunto con las unidades de almacenamiento y las redes de datos, para que el servicio de tipo cloud pueda escalar los recursos fácilmente. En particular, la unidad estándar de virtualización se denomina *máquina virtual* y permite encapsular un sistema operativo dedicado a cómputo, almacenamiento o tráfico de red en una entidad independiente. Cada una de estas unidades independientes se denomina *nodo*. En una arquitectura de procesamiento distribuido se utilizan varios nodos de cómputo, donde cada nodo es un servidor físico o virtualizado dedicado a la ejecución de la aplicación. Este mismo concepto se utiliza además en almacenamiento distribuido, donde cada nodo de la red de almacenamiento corresponde a un servidor de bases de datos. Según se describe en el artículo de Grossman, existen dos tipos de servicios de tipo cloud: aquellos que proveen recursos de cómputo a demanda (como es el caso de Amazon EC2, <http://aws.amazon.com/ec2/>), que permiten escalar en cantidad de computadores; y aquellos que proveen capacidad a demanda (como es el caso de Google Compute Engine, <https://cloud.google.com/compute/>) donde se provee poder de procesamiento total independientemente del número de computadores asignados. Estos dos tipos de computación cloud dan lugar a diferentes modelos de servicio de acuerdo al tipo de recurso que se provee en cada caso.

Los principales modelos de servicio en el paradigma de cloud computing son SaaS (Software as a Service), IaaS (Infrastructure as a Service) y PaaS (Platform as a Service) ([Mell y Grance, 2011](#)). El modelo SaaS define una plataforma de servicios en la que el proveedor dispone de un conjunto de aplicaciones accesibles por usuarios del servicio. Por su parte, el modelo PaaS define una plataforma de servicios donde se provee un entorno de desarrollo para que el usuario pueda construir y ejecutar su aplicación sin preocuparse por cuestiones tales como el sistema operativo y las bases de datos. Finalmente, el modelo IaaS define una plataforma de servicios donde lo único que se provee son recursos computacionales virtualizados, por lo que el usuario es el responsable de la construcción y la ejecución de aplicaciones y todo su contexto. Los límites entre modelos no están tan claros en la implementación por parte de los diferentes proveedores, y es común que se provean aspectos de cada modelo en las diferentes plataformas. La Figura [3.1](#) presenta la organización de los tipos de cloud en términos de dependencia entre ellas. La capa de servicios de tipo IaaS provee los recursos computacionales fundacionales

como servidores y almacenamiento. Sobre la capa IaaS se construye la capa de tipo PaaS que provee herramientas de desarrollo, bases de datos y motores de ejecución de aplicaciones. Finalmente, la capa de tipo SaaS aprovecha los recursos provistos por la capa de tipo PaaS para proveer servicios al usuario final, como aplicaciones de correo electrónico, manejadores de contenidos, etc.

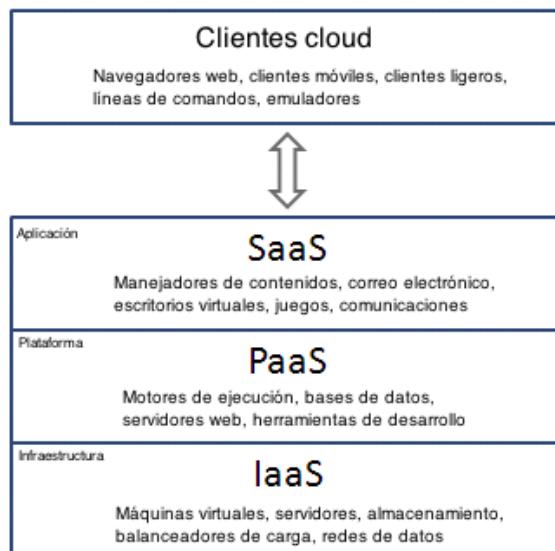


Figura 3.1: Modelos de servicio cloud organizados como capas en una pila (imagen de dominio público tomada de Wikipedia y traducida al español)

En general, el costo de uso de un servicio cloud está asociado a la cantidad de recursos efectivamente utilizados. La capacidad de agregar recursos en tiempo de ejecución se denomina *elasticidad*. Se dice que un sistema es elástico si es capaz de agregar recursos computacionales a medida que se necesitan, así como de liberarlos cuando ya no se utilizan. Esta es una característica fundamental para asegurar a procesos que pueden demorar varios días la disponibilidad de recursos computacionales. Los recursos que componen el sistema elástico pueden ser de diversos tipos, como ser servidores para el procesamiento computacional, servidores de bases de datos, colas de mensajes, etc. Otra característica fundamental de los servicios de tipo cloud es la *resiliencia*. La resiliencia define la capacidad de un componente de recuperarse de una falla sin perjudicar el funcionamiento del sistema. En cloud computing se logra la resiliencia a través de la redundancia de recursos, como por ejemplo múltiples servidores ejecutando la misma aplicación o la duplicación de datos en varias máquinas. En el caso de los servidores que ejecutan una aplicación, se establece al menos otro servidor espejo capaz de replicar el mismo comportamiento del

servidor original en caso de problemas. En el caso de la duplicación de datos se realizan copias de la información con la que opera una determina aplicación en más de un servidor, para mantener el comportamiento del sistema en caso que el servidor principal sufra alguna falla. Un sistema que es resiliente y que además es elástico, posee las características necesarias para lograr lo que se denomina alta disponibilidad (high availability). El concepto de alta disponibilidad es fundamental para el diseño de una arquitectura de procesamiento distribuido, debido a que asegura que todos los componentes de la arquitectura ejecuten sus tareas y no se vean afectados por el resto. Mediante la elasticidad se asegura la cantidad de recursos suficientes para llevar a cabo la tarea de cada componente de la arquitectura distribuida y mediante la resiliencia se asegura un mecanismo de respuesta a fallos que permite continuar con la ejecución a pesar de los problemas que puedan surgir ([Erl et al., 2013](#)).

Existen ciertas desventajas en el modelo de cloud computing que conviene analizar de acuerdo al contexto de la aplicación que se está construyendo. Si se utiliza un proveedor externo, los servicios estarán ubicados en servidores que no pertenecen al usuario o a la organización que los está utilizando. La ubicación de los servicios en servidores remotos repercute en los tiempos de acceso y en la disponibilidad de los recursos computacionales. Por ejemplo, pueden existir tiempos de espera en la asignación de servidores como recursos de un sistema elástico que pueden afectar de forma aleatoria (y difícil de medir) los tiempos estimados de ejecución de las tareas de cómputo. El sistema puede ser elástico, pero también puede no asignar de los recursos en forma inmediata a la aplicación que los solicita. Además, en caso de que los recursos físicos no estén aislados por usuario, se corren riesgos derivados del uso compartido de los recursos. Cualquier problema de hardware ocasionado por alguna de las tareas afecta a las demás indefectiblemente, ya que el entorno suele estar virtualizado para permitir utilizar la misma máquina física en procesos de diferentes usuarios.

Finalmente, se presenta el problema de la privacidad. En el caso de la utilización de un proveedor de infraestructura de tipo cloud, la información de contexto de ejecución de la aplicación no está enteramente bajo el control del usuario. Esta información puede consistir no solo del código de ejecución, sino además de información sensible relativa a las tareas que se llevan a cabo. El manejo de información sensible por parte de terceros representa una responsabilidad especial en la definición de políticas de privacidad por parte de la

organización que provee el servicio, y una evaluación de los riesgos por parte de los usuarios.

3.2. Computación de alto desempeño

Esta sección presenta los modelos de programación de alto desempeño, los modelos de computación paralela y la utilización de ambos en este trabajo.

3.2.1. Modelos de computación de alto desempeño

Según [Foster \(1995\)](#), en el campo de la computación se puede hablar de diferentes paradigmas, como la computación centralizada, la computación distribuida y la computación paralela. La computación centralizada corresponde al modelo clásico de procesamiento de información utilizando una infraestructura conformada por una computadora, memoria y disco en un mismo lugar físico. El procesamiento se lleva a cabo en una ubicación central a la cual se accede mediante terminales conectadas al servidor central. Por otra parte, el paradigma de computación distribuida define un conjunto de recursos separados físicamente pero interconectados mediante algún sistema de comunicación (red de área local o internet, por ejemplo). Este diseño permite resolver problemas que requieren cómputo intensivo, ya que permite agregar o quitar recursos computacionales según sea necesario. La capacidad de distribuir el cómputo está soportada por el diseño de arquitectura distribuida. Si el programa que ejecuta el procesamiento puede ejecutar en varios recursos de cómputo simultáneamente se denomina *programa paralelo*. La motivación principal detrás del paradigma de computación paralela es poder procesar grandes volúmenes de datos, implementar modelos complejos y resolver problemas con órdenes de complejidad grandes.

3.2.2. Modelos de computación paralela

Para diseñar un programa paralelo se consideran dos alternativas: paralelismo a nivel de datos y paralelismo a nivel de control. El paralelismo a nivel de datos define un paradigma que propone la utilización de múltiples procesos ejecutando el mismo código de aplicación sobre conjuntos diferentes de datos con el fin de mejorar el tiempo de procesamiento del conjunto total de datos

con respecto al tiempo del programa secuencial. Por otra parte, el paralelismo a nivel de control define un paradigma donde se propone mejorar el tiempo de ejecución del programa secuencial mediante la división del código en secciones que realizan tareas diferentes sobre los mismos datos.

La Figura 3.2 presenta un ejemplo de utilización del paralelismo a nivel de datos para el procesamiento de archivos de entrada. Dado un conjunto de archivos de tamaño M , se divide el conjunto en N subconjuntos disjuntos y se asigna la tarea de procesamiento de cada uno de estos subconjuntos a un procesador distinto. Cuando los N procesadores finalizan las tareas de procesamiento de los archivos se obtiene un conjunto de resultados que corresponde al total de los archivos de entrada. La consecuencia de la simultaneidad del procesamiento de los archivos tiene como consecuencia la reducción del tiempo de procesamiento total del conjunto de archivos con respecto a una eventual ejecución secuencial.

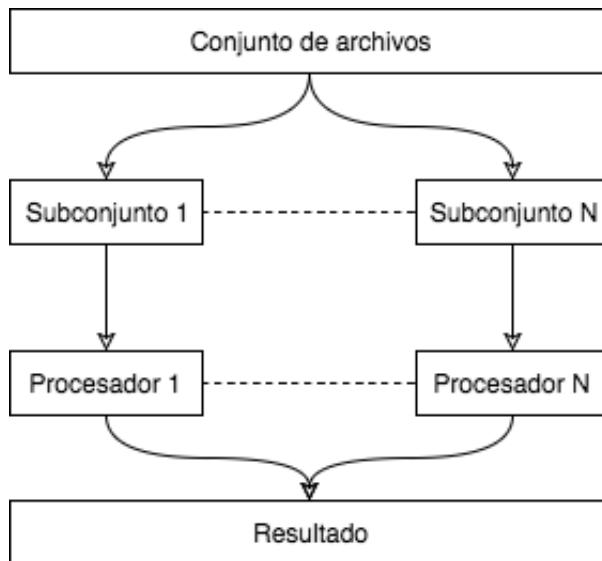


Figura 3.2: Aplicación del paralelismo a nivel de datos al procesamiento de archivos

La Figura 3.3 presenta un ejemplo de utilización del paralelismo a nivel de control para el procesamiento de un archivo de entrada (en este caso una imagen). Se desea aplicar diferentes filtros a la imagen, donde cada filtro es independiente de los demás. El código de cada filtro se ejecuta en un procesador diferente y se copia la imagen de entrada a cada uno de estos procesadores. Como resultado, se obtienen N versiones de la imagen correspondientes a los N filtros aplicados por separados. En un paso final se unifica la información

de cada uno de estas versiones de la imagen para dar como resultado un solo archivo de salida con el contenido final equivalente a una combinación de los filtros determinada por el código de la aplicación.

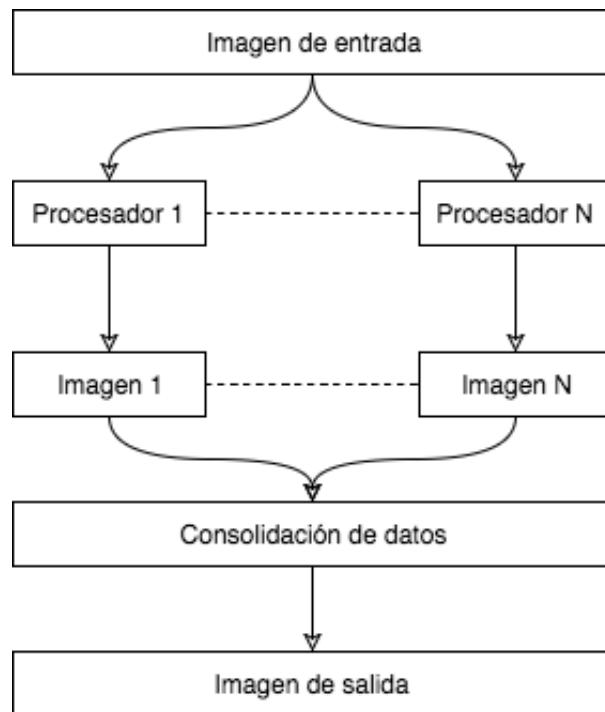


Figura 3.3: Aplicación del paralelismo a nivel de instrucción al procesamiento de una imagen

3.2.3. Computación de alto desempeño y computación paralela en el problema del análisis masivo de imágenes astronómicas

En el software desarrollado en esta tesis se utilizan los conceptos de computación distribuida y de paralelismo a nivel de datos. La infraestructura necesaria para la distribución de las tareas de cómputo está dada por un proveedor de tipo IaaS/PaaS, mientras que el paralelismo a nivel de datos se establece mediante el procesamiento de una imagen por vez en cada servidor. El proveedor de tipo IaaS/PaaS es Microsoft Azure y se utiliza para proveer las máquinas virtuales que ejecutan el código de la aplicación. En el esquema definido en Azure, cada máquina virtual procesa una imagen astronómica a la vez hasta que la totalidad de las imágenes son procesadas. El paralelismo a nivel de

datos consiste en la ejecución en forma paralela del código de la aplicación de procesamiento de imágenes astronómicas en varias máquinas virtuales en forma simultánea, lo que permite minimizar el tiempo total de procesamiento del conjunto total de imágenes. Los detalles de esta arquitectura se presentan en el Capítulo 5.

3.3. Almacenamiento, replicación y disponibilidad de los datos

La arquitectura de datos es el área que estudia el diseño de sistemas en torno a las características de los datos que se quieren analizar. En este trabajo se propone diseñar un sistema que posea algunas características específicas de la arquitectura en torno a los datos como ser la replicación (Son, 1988) y la alta disponibilidad, con el propósito de garantizar el acceso a las imágenes astronómicas desde los nodos que las requieran. Garantizar el acceso a los datos es clave en un sistema distribuido como el que se construye en este trabajo, donde se plantea un uso intensivo de los datos en un período de tiempo prolongado. Con el fin de obtener valores experimentales para las métricas de evaluación del desempeño, que permitan evaluar correctamente los algoritmos desarrollados y las arquitecturas estudiadas, es necesario minimizar el tiempo de acceso a los archivos que contienen la información que se procesa en cada nodo y esto se logra garantizando el acceso a los datos en todo momento. Como se explica en el trabajo de Son (1988), una forma de garantizar el acceso a los datos consiste en mantener copias de la información en varios servidores. Estos servidores se denominan *réplicas* porque contienen en principio la misma información que el servidor principal, aunque se han propuesto esquemas que permiten distribuir la replicación de datos de acuerdo a otras reglas. En la Figura 3.4 se presenta la estrategia de replicación de datos basada en el diseño de arquitectura que se denomina master-slave. En una arquitectura de tipo master-slave existe un servidor que es la autoridad respecto a los datos y existen otros servidores que pueden responder pedidos de lectura realizados por parte de aplicaciones de usuario. Los pedidos de escritura recaen exclusivamente sobre el servidor principal.

En la Figura 3.5 se presenta una solución alternativa al esquema master-slave donde se utilizan múltiples servidores con información replicada y sin

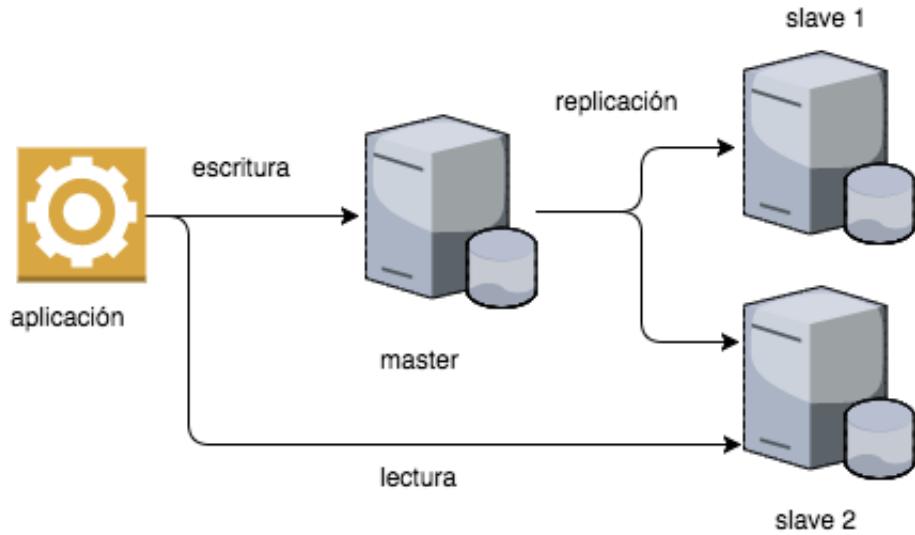


Figura 3.4: Arquitectura de tipo master-slave para la replicación de datos

un servidor principal identificado. En general, una arquitectura de servidores interconectados donde todos cumplen una función similar se denomina *cluster*, y cada uno de los servidores se denomina *nodo* (Yeo *et al.*, 2006). La conexión entre servidores se presenta en forma de anillo porque ningún servidor tiene mayor importancia que el resto, debido a que el esquema de asignación y recuperación de los datos está presente en cada nodo por igual y los datos se distribuyen a lo largo del cluster en base a lo que se denomina clave de particionamiento (partition key). La clave de particionamiento forma parte del diseño de la arquitectura de los datos y es muy importante considerar varios aspectos en su elección de acuerdo a qué atributos de los datos se utilizarán para el particionamiento. Si la clave de particionamiento está correctamente elegida, la carga se distribuirá de forma uniforme sobre los nodos, pero si se elige una clave que no distribuye apropiadamente la carga se corre riesgo de sobrecargar uno de los nodos. Cuando se quiere recuperar un dato del cluster la lectura se realiza sobre cualquiera de los nodos, ya que en una arquitectura de este tipo todos los nodos son responsables de conocer el estado del resto de los nodos del anillo. Si el nodo que recibe el pedido de consulta (de tipo lectura) tiene el dato buscado la búsqueda finaliza y se retorna el valor correspondiente, pero si la clave de particionamiento del dato no corresponde al nodo consultado, de acuerdo al esquema de particionamiento diseñado, se obtiene una lista de los nodos que pueden responder a la consulta. Por ejemplo, en una arquitectura de tipo anillo de 5 nodos se puede definir un esquema de replicación de 3 nodos,

que implica que no todos los nodos tienen la misma información, pero por cada archivo existen dos copias en otros nodos. La cantidad de nodos del cluster que mantienen una copia de los datos se denomina *factor de replicación* y es un valor de configuración presente en los servidores. En el caso de que uno de los nodos deje de estar operativo por algún motivo (por ejemplo una falla en el servidor o por algún problema de comunicación en la red de datos), la información permanece disponible en los nodos del cluster designados como réplica del nodo no operativo. La estrategia de replicación de datos de un cluster de servidores depende fuertemente de aspectos tales como la localización física de los servidores y la confiabilidad de la red de comunicaciones entre los mismos. Si cualquiera de los nodos deja de estar operativo se asegura la disponibilidad y accesibilidad de los datos. En el momento en que un nodo sufre una falla, se pueden tomar dos decisiones: reemplazar el nodo caído con uno nuevo; o redistribuir las particiones para mantener el factor de replicación configurado. Este comportamiento ante la falla de un nodo forma parte de la configuración de la aplicación que ejecutan los servidores del cluster.

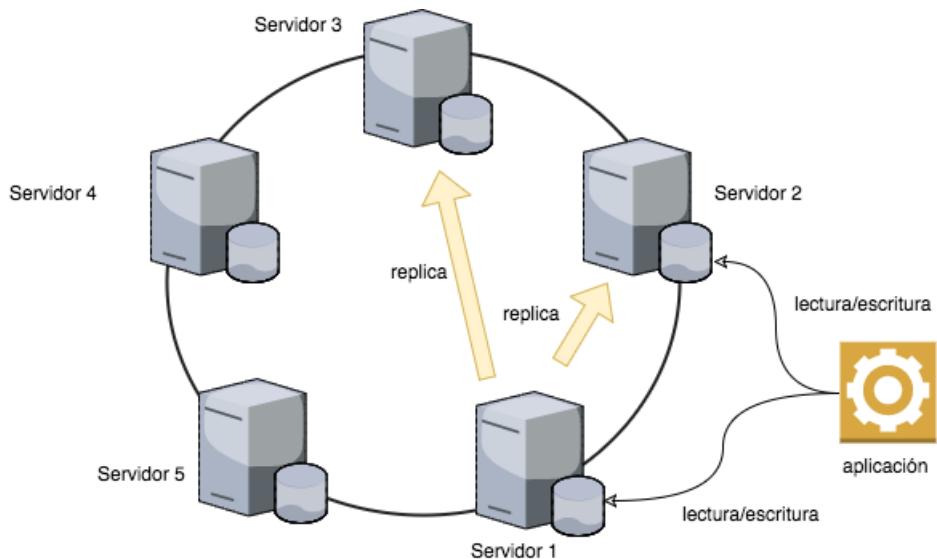


Figura 3.5: Arquitectura de tipo cluster con forma de anillo para la replicación de los datos

Los esquemas de replicación de datos pueden ser aplicados en diferentes tipos de bases de datos. Por ejemplo, se pueden particionar y replicar bases de datos relacionales, bases de datos no relacionales, blob storage (datos binarios sin estructura predefinida), file storage (archivos), etc. El mismo principio

principio de evitar el punto único de falla es válido en todos los escenarios considerados y es un factor clave para garantizar la disponibilidad de los datos en todo momento para que la capa de aplicación pueda realizar transacciones.

Capítulo 4

Trabajos Relacionados

El tratamiento y análisis de información relacionada al espacio es un área que no ha estado ajena al advenimiento de las nuevas técnicas de computación en el siglo 21. En particular, en el mismo año que se divulgó la técnica de MapReduce ([Dean y Ghemawat, 2008](#)) se propusieron los primeros trabajos que involucraron la aplicación de la técnica a flujos de datos científicos. Como ejemplo de trabajo pionero, [Mackey et al. \(2008\)](#) plantearon un análisis de los diversos componentes que integran Hadoop y cuáles de ellos pueden aplicarse directamente al tratamiento de información en tres dominios diferentes (astrofísica, bioinformática y ciberseguridad). Por fuera del contexto de la industria de software, se detectó el interés de la comunidad científica por adoptar el uso de las nuevas formas de procesamiento de información provenientes de esa industria. Como dato final, en el trabajo de [Mackey et al. \(2008\)](#) se llegó a la conclusión que la base de datos distribuida de Hadoop (HDFS) supone un gran avance frente al almacenamiento tradicional provisto por bases de datos relacionales y que además la estructura de MapReduce se ajusta muy especialmente a flujos de tipo simulación-persistencia-procesamiento, como los que habitualmente se utilizan para el procesamiento de imágenes astronómicas.

Más cercano en el tiempo, y en la misma dirección del trabajo de [Mackey et al., 2008](#), se encuentra el análisis comparativo que propuso [Gunarathne et al. \(2010\)](#). En este trabajo se establece la dualidad de formas de ejecutar un proceso del tipo MapReduce en las plataformas disponibles en cloud. El análisis comparativo propuesto se puede dividir en dos partes: trabajar con una infraestructura ya existente que facilita la planificación y el procesamiento de trabajos o por el contrario establecer una arquitectura propia para tal fin.

La motivación principal de este análisis comparativo es establecer la posibilidad de pensar aplicaciones fuertemente basadas en entrada y salida de datos, que posean la capacidad de mantenerse consistentes y tolerantes a fallos en un entorno de procesamiento distribuido, y demostrar que la performance de estas aplicaciones puede alcanzar y mejorar a las implementaciones en clusters dedicados. A pesar que la conclusión y las comparativas finales obtenidas por [Gunarathne et al.](#) están basadas fuertemente en un framework construido a medida (denominado AzureMapReduce) se pueden extraer conceptos importantes de su trabajo, en particular sobre el análisis acerca de los problemas a resolver cuando se decide utilizar MapReduce implementado sobre una plataforma de servicios cloud. Los principales conceptos establecidos en su análisis son:

- Almacenamiento de datos: en general se dispone de tres alternativas respecto al almacenamiento de datos: Off-Instance Cloud Storage (S3, Azure Blob Storage, etc.), Off-Instance Block storage (Amazon Elastic Block Storage, o EBS) y Virtualized Instance Storage (que en este caso se utiliza para la implementación de HDFS). La elección del tipo de almacenamiento en cloud es fundamental para lograr que la performance de la aplicación sea óptima al momento de recuperar los datos, considerando también que el ancho de banda sea el adecuado.
- Almacenamiento de metadatos: para el correcto funcionamiento del sistema de planificación de tareas y la sincronización de datos entre las instancias es necesario el intercambio de metadatos sobre los trabajos de procesamiento que se están realizando. Es fundamental que este almacenamiento no se transforme en un cuello de botella y escale correctamente según las necesidades del ambiente de ejecución de la aplicación.
- Escalabilidad y consistencia en la comunicación: este punto refiere estrictamente a las operaciones del tipo entrada/salida y debe entenderse como una necesidad de tener claro un defecto que muestran los servicios de tipo PaaS, donde es posible que haya fluctuaciones causadas por la red que intercomunica a los nodos (ninguna solución que se construya tendrá conocimiento absoluto de la infraestructura a nivel del proveedor). Este problema afecta directamente a la transferencia de datos entre los nodos.
- Consistencia en la performance: relacionado con el concepto anterior,

pero visto desde otro punto de vista. Los nodos son procesos que ejecutan en máquinas virtuales que están, a su vez, ejecutando en máquinas físicas del datacenter del proveedor del servicio. Estas máquinas físicas pueden estar compartidas con otros usuarios a los cuales el proveedor les asignó el mismo hardware subyacente para la ejecución de sus aplicaciones. Trabajar en un entorno compartido puede provocar una degradación en los servicios de acuerdo a la carga del procesador o a la utilización de la red en cada máquina.

- Confiabilidad (fallas en los nodos): en general, cualquier aplicación que requiera de muchos nodos de cómputo para procesar trabajos de procesamiento puede sufrir eventualmente la falla de alguno de los nodos. En particular, es probable que una falla en el funcionamiento de un nodo suceda con más frecuencia en arquitecturas de cloud computing basadas en máquinas físicas compartidas entre muchas aplicaciones. A pesar que el algoritmo de MapReduce anticipa la falla de cualquiera de los nodos y provee mecanismos de soporte ante estas situaciones, la falla definitiva del nodo maestro significaría una catástrofe.
- Elección del tipo de instancia (o máquina virtual): es un detalle de configuración de la arquitectura que está directamente relacionada a las opciones que provea la plataforma que se utilice como servicio. De todas maneras es necesario tener en cuenta las características del procesador que se requiere para la ejecución del procesamiento así como la disponibilidad de memoria para la correcta ejecución de la aplicación. Ambas configuraciones deben ser las adecuadas para garantizar la performance que se quiere lograr y al mismo tiempo se deben optimizar los costos asociados al uso de estos recursos.
- Registro de datos: en principio los nodos de cómputo destruyen cualquier información local una vez que finalizan su ejecución, por lo que es importante tener mecanismos para preservar la información crítica en algún otro tipo de almacenamiento.

Como antecedente importante, [Alonso-Calvo *et al.* \(2010\)](#) presentaron un problema muy similar al tratado en este trabajo, con interesantes puntos de contacto. La problemática del trabajo de [Alonso-Calvo *et al.*](#) consiste en alma-

cenar de forma distribuida imágenes de gran tamaño procedentes de instrumentos espaciales y facilitar su procesamiento en un esquema de tipo multi-agente. Este problema puede dividirse en tres grandes partes:

1. El almacenamiento y la distribución de los datos: las imágenes deben estar contenidas en bases de datos distribuidas en cloud, que permita el acceso por parte de todos los agentes que estén realizando tareas sobre ellas. En particular, se propone la estrategia de dividir y almacenar regiones de las imágenes extremadamente grandes, utilizando una técnica de representación de datos mediante grafos. Mediante un algoritmo desarrollado especialmente para la tarea de división, cada imagen es analizada de acuerdo a un patrón de píxeles y dividida en base a criterios relacionados con la densidad de colores. La imagen a analizar es asignada a un agente encargado de dividirla en regiones, y cada región de esta imagen es almacenada y asignada a otros agentes con poca carga para que se encarguen del procesamiento parcial de cada región.
2. La selección de datos y asignación de tareas: para la asignación inicial de tareas, se cuenta con un servidor denominado Resource Index. Este servidor contiene información sobre todas las tareas programadas en cada agente y una estimación del tiempo que llevará cada tarea. La información acerca del estado de cada tarea es crucial para el balanceo de carga, esto es, para determinar cuáles agentes son los menos ocupados y poder así hacer una asignación equilibrada de tareas a recursos.
El Resource Index contiene información que es brindada por los propios agentes mediante el planificador local de cada uno. Cuando una tarea es asignada a un agente, el proceso de planificación determina el tiempo estimado que llevará la tarea y la coloca en una cola de procesos. Inmediatamente, el planificador notifica al Resource Index la información relacionada a dicha tarea, y así sucesivamente para cada tarea en cada agente. En particular, el Resource Index es consultado por los agentes que dividen las imágenes para obtener los agentes con menos carga y así asignar el procesamiento de las regiones de las imágenes con la mayor eficiencia posible.
3. El procesamiento y los resultados: el procesamiento en cada agente, una vez desencolado el trabajo localmente y obtenidos los datos de la tarea, se realiza sobre la estructura almacenada en la base de datos. El algoritmo

de división que se desarrolla en el trabajo de [Alonso-Calvo et al.](#) permite unir los resultados de forma coherente de manera de obtener datos por cada imagen como si hubiera sido procesada de una sola vez. Lo más interesante de los resultados obtenidos por [Alonso-Calvo et al.](#), más allá de la información puntual que se obtuvo de las imágenes, es la ventaja de implementar un procesamiento distribuido a nivel de datos. Se tomaron seis imágenes bien conocidas y se utilizaron como entrada del sistema construido. El aumento de la cantidad de agentes (de dos a tres) para procesar las diferentes regiones de cada imagen, a través de la asignación mediante el algoritmo de balanceo y distribución, permite mejorar los tiempos a medida que crece la capacidad de procesamiento. En particular se logra reducir el tiempo de ejecución en un 51,22 % utilizando dos agentes y en un 64,50 % utilizando tres agentes. Se obtiene además una mejora en el tiempo de dilatación morfológica de la imagen en un 39.86 % y en un 53,54 % utilizando dos y tres agentes, respectivamente.

Otro antecedente significativo y muy relacionado con la problemática que se trata en esta tesis, es el trabajo de [Li et al. \(2010\)](#). Dado que tiene muchos puntos en común con el problema del tratamiento de datos científicos y fue implementado sobre Microsoft Azure este trabajo se comenta en detalle a continuación.

La motivación del trabajo de [Li et al.](#) es implementar un mecanismo de procesamiento de información proveniente de múltiples fuentes, todas relacionadas y pertenecientes al mismo dominio, aunque con significados diferentes. Los datos provienen del proyecto MODIS (del inglés, Moderate Resolution Imaging Spectroradiometer), que nutre a la comunidad científica con datos captados por dos satélites específicos: Terra y Aqua. Estos dos satélites recogen diferentes métricas relacionadas al planeta y a su atmósfera, y actualizan cada dos días la base de datos disponible de manera que los investigadores cuentan con información cruda lista para procesar de manera continua. Dado el tamaño de los archivos y la cantidad de entradas que se han generado a lo largo del tiempo, se han considerado diversas alternativas para facilitar lo que se denomina “reproyección” de la información en otros formatos más cómodos para su manejo y consulta. En particular, el trabajo de [Li et al.](#) propone una arquitectura en cloud para ir procesando la información y reproyectando a medida que se solicita (a través de una interfaz web diseñada con tal

propósito). La arquitectura proporciona la ventaja de disponer de una gran cantidad de instancias de tipo worker para realizar el trabajo y almacenamiento compartido que permite evitar trabajo duplicado. Una vez que un investigador solicita una determinada reproyección de datos, el resultado queda disponible para otros investigadores de manera inmediata y sin necesidad de tener que calcular la misma reproyección en el futuro. La Figura 4.1 presenta la arquitectura propuesta por [Li et al.](#) utilizando MODIS y Azure.

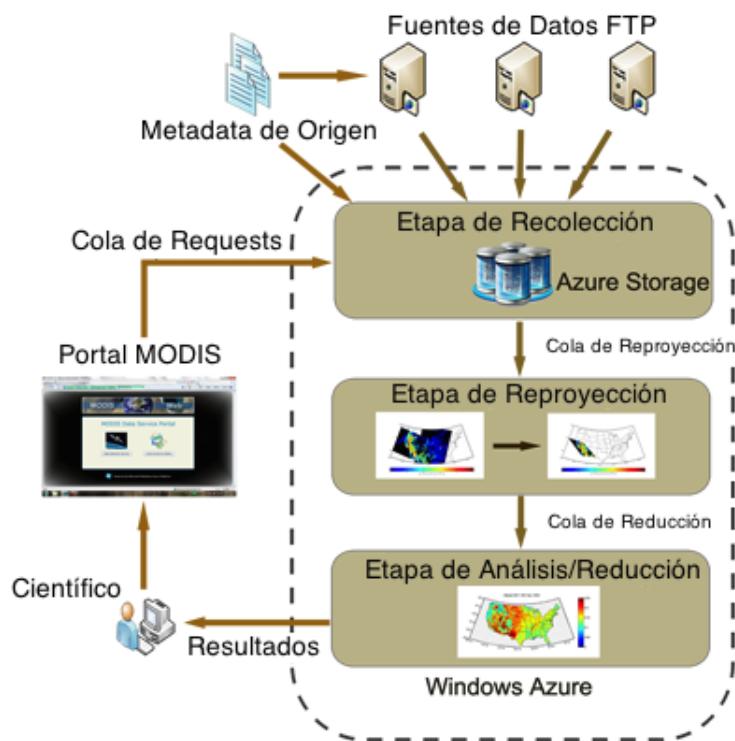


Figura 4.1: Arquitectura de la solución (imagen con permiso de reproducción, tomada de [Li et al. \(2010\)](#) y traducida al español)

La arquitectura propuesta por [Li et al.](#) es simple pero efectiva. La Figura 4.2 presenta el diseño del planificador y su interacción con los diferentes componentes de Azure. La escalabilidad se asegura a través de la cola de mensajes que permite desacoplar la asignación de trabajos respecto a las máquinas que van a realizarlos. Mediante la interfaz web se permite a los investigadores elegir el tipo de consulta a realizar (lo que se traduce en las reproyecciones necesarias) y se almacenan los resultados en la base de datos de tipo blob storage. También se almacenan entradas en una

base de datos de tipo relacional que representan el estado de los trabajos así como la constancia de reproyecciones que ya han sido realizadas. Esta última funcionalidad es clave al momento de reducir la utilización de recursos.

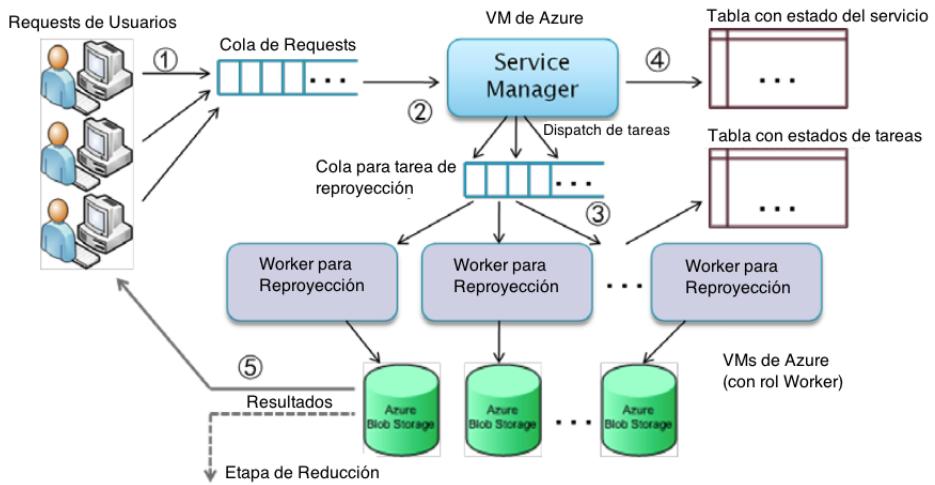


Figura 4.2: Diseño del planificador (imagen con permiso de reproducción, tomada de [Li et al. \(2010\)](#) y traducida al español)

Existe de todas maneras una problemática en el diseño que proponen los autores. En principio la arquitectura es escalable y pueden agregarse tantas máquinas como se deseen e iniciar la ejecución de tantos trabajos como se requiera. Pero existe un problema con el acceso a los datos de forma concurrente. Dado que no se modifican las fuentes de información no hay problemas de concurrencia, pero el acceso simultáneo de muchos procesos a los mismos archivos o blobs puede significar un aumento de latencia y por consiguiente una pérdida de performance global como efectivamente se reporta en los resultados de [Li et al.](#): la performance disminuye. El tiempo total de comunicación al ejecutar 1500 trabajos de reproyección aumenta de 20 horas a 27 horas al pasar de 50 nodos de cómputo a 150 nodos de cómputo. Esta degradación de performance es un síntoma sobre las posibles limitaciones del blob storage de Azure, y es un aspecto a considerar seriamente ya que podría significar una pérdida de toda la performance ganada a través de la escalabilidad a nivel de procesamiento. En general, se pueden pensar arquitecturas que contemplen almacenamiento distribuido o basado en localidad de datos que asegure

que el acceso no va a ser concurrente siempre sobre los mismos nodos de almacenamiento, o bien se puede partir a nivel de datos cada blob y dejarlo en cache para que varios procesos puedan trabajar sobre él. La estrategia de almacenamiento depende de qué tan deficiente resulte la escalabilidad en este punto.

Finalmente, a la hora de evaluar las diferentes plataformas y tecnologías que se ven involucradas en la implementación de una solución de procesamiento distribuido de información científica, [Dobre y Xhafa \(2013\)](#) presentaron un panorama relacionado a computación en cloud y HPC en los últimos años, junto con los desafíos que cada etapa planteó, desde la refundación de paradigmas de programación distribuida y paralela en la nueva era de Big Data hasta frameworks que facilitan la implementación de MapReduce y sus variantes.

En esta misma línea, el trabajo de [Parashar *et al.* \(2013\)](#) profundiza en las diferentes formas de interacción entre HPC y Cloud Computing, considerando tres alternativas: HPC en cloud, HPC más cloud y HPC como servicio. Si bien el trabajo de [Parashar *et al.*](#) está centrado en evaluar la aplicabilidad y performance de un framework en particular (CometCloud) bajo cada alternativa, es interesante entender el tipo de abstracción y la arquitectura de la solución que se considera para la implementación de la propuesta de los autores.

Como ya se ha mencionado, una de las principales problemáticas que surgen al momento de considerar datos acumulados a lo largo de décadas, y en particular imágenes procedentes de telescopios, es la transferencia y manejo de un gran volumen de datos. [Szabo *et al.* \(2013\)](#) argumentaron que el problema de manipular un gran volumen de datos para la ejecución de workflows científicos en cloud es de tipo NP completo. Para resolver el problema, [Szabo *et al.*](#) propusieron un heuristic en base a algoritmos genéticos que permite reducir los tiempos de transferencia inherentes al manejo de datos en cloud para la ejecución de workflows de análisis científico.

Capítulo 5

Arquitectura propuesta

Este capítulo presenta diferentes opciones de arquitecturas escalables que permiten procesar el volumen de datos requerido por el proyecto “Geophysics using Hubble Space Telescope” ([Tancredi *et al.*, 2016](#)) en tiempo razonable. Complementariamente, se evalúan las tecnologías aplicadas a las diferentes arquitecturas consideradas. En la sección [5.1](#) se propone una arquitectura basada en contenedores Docker administrados por Mesos y en la sección [5.2](#) se presenta la solución final, adaptada a la infraestructura disponible en Microsoft Azure, con los scripts de procesamiento desarrollados en Python.

5.1. Arquitectura utilizando Apache Mesos y Marathon

En esta sección se presenta la arquitectura utilizando Mesos y Marathon. Se describen los detalles técnicos para la ejecución de las tareas de procesamiento y mejoras en el algoritmo de planificación de estas tareas.

5.1.1. Pipeline de procesamiento de imágenes utilizando Mesos y Marathon

Como parte de la investigación inicial de este trabajo se consideró el enfoque de encapsular el código existente y aplicar el principio de caja negra. En el principio de caja negra se utiliza un módulo del cual se desconoce a priori el contenido, pero se entienden las interacciones de entrada y salida ([Bhasin *et al.*, 2014](#)). En una primera etapa, se distribuyó el módulo de tipo caja ne-

gra que contiene el código del procesamiento de las imágenes astronómicas a diferentes nodos de cómputo. De esta manera se logró un paralelismo a nivel de datos como se definió en el Capítulo 3 y se obtuvieron resultados iniciales de eficiencia computacional, que se utilizan para comparar con los resultados de las ejecuciones sobre la arquitectura que se describe en la Sección 5.2. Para poder utilizar el código existente en los diferentes nodos de cómputo, el primer paso es crear un script de instalación que provea el ecosistema necesario para la ejecución del paquete IRAF y luego desarrollar una arquitectura que permita utilizar en cada nodo el script de configuración de ambiente y la lista de imágenes a procesar. En particular, para el encapsulamiento del código IRAF CL se utilizó Docker ([Docker Inc., 2016](#)), una plataforma que permite crear y distribuir aplicaciones sobre un sistema operativo virtualizado de forma sencilla y ligera en forma contrapuesta a lo que podría representar una máquina virtual tradicional. El Código 5.1 muestra la configuración del contenedor que fue utilizado para instalar y ejecutar el paquete de análisis de imágenes astronómicas a partir de un sistema operativo Ubuntu.

```
FROM ubuntu:14.04
MAINTAINER German Schnyder <gschnyder@gmail.com>
ENV DEBIAN_FRONTEND noninteractive
ENV SHELL /bin/bash
RUN apt-get update
RUN apt-get install -y <PACKAGES>
RUN mkdir -p /usr/local/workingdir
WORKDIR /usr/local/workingdir
RUN wget http://ssb.stsci.edu/ssb_installer
RUN chmod +x ssb_installer
RUN printf "ssbdev\n\n" | ./ssb_installer
```

Código 5.1: Ejemplo de declaración de contenedor Docker

Una vez definido el entorno de ejecución para el procesamiento de una imagen, es necesario establecer el mecanismo de distribución genérico de imágenes a cada nodo. Considerando los recursos computacionales con los que se contaba en la fase de investigación (un servidor dedicado y un cluster compartido, como se detalla en el Capítulo 6), se decidió utilizar Apache Mesos ([Hindman et al., 2011](#)). Apache Mesos permite abstraer los recursos de cómputo disponibles de forma transparente al lugar físico en donde se encuentren, por lo que se puede trabajar con uno o muchos servidores de cómputo a la vez de forma transpa-

rente a la lógica de ejecución de las tareas. Una ventaja particular de Mesos es que se integra de forma nativa con Docker, por lo que se pueden crear tareas en Mesos que instancien un contenedor Docker e inicien su ejecución a través de la especificación de determinados parámetros de entrada. Existen alternativas a Mesos (como Kubernetes, Swarm, Fleet, etc.) que también se integran con Docker y cualquiera de ellas representa una alternativa válida para la discusión sobre la arquitectura necesaria para el procesamiento de las imágenes. En una posible línea de trabajo futuro se podría realizar una evaluación de las diferentes alternativas a Mesos que permiten aprovechar recursos heterogéneos y ejecutar código encapsulado en contenedores Docker.

Por otra parte, para la definición de tareas se utiliza Marathon [Mesosphere Inc. \(2016\)](#). Marathon funciona como orquestador de recursos y tareas que serán ejecutadas en Mesos. Mediante un archivo de configuración se definen los requerimientos de las ejecuciones como se muestra en el Código 5.2. Como ya se mencionó, el soporte para Docker es nativo en el ecosistema de tecnologías asociadas a Apache Mesos, por lo que la configuración de Marathon puede incluir referencias específicas de este tipo de contenedor. En particular, el contenedor diseñado para el procesamiento de las imágenes en este trabajo es almacenado en Dockerhub (<https://hub.docker.com/>), el repositorio oficial de imágenes Docker. Como parte del contexto de ejecución de un contenedor se indica el comando de arranque, o en su defecto la ubicación del script de arranque, para comenzar con el procesamiento. Un ejemplo de script de arranque se muestra en el Código 5.3. De manera similar a la configuración necesaria para Docker, como parte de los parámetros de entrada de Marathon se especifica un identificador propio de Dockerhub que indica qué contenedor se quiere utilizar. Además, en la configuración enviada a Marathon se establecen las pre-condiciones de ambiente que requiere la tarea para su ejecución, a saber: memoria, disco, tiempo de procesador y cantidad de instancias.

```
{  
    "id": "iraf", --identificador de la tarea  
    "cmd": "comando", --comando para iniciar el procesamiento  
    "cpus": 0.2, --porcentaje de CPU asignado  
    "mem": 20.0, --memoria asignada  
    "instances": 1, --instancias totales  
    "constraints": [], --restricciones de contexto  
    "container": {
```

```

    "type": "DOCKER", --tipo de contenedor
    "docker": {
        "image": "gschnyder/iraf-worker" --Dockerhub ID
    }
}
}

```

Código 5.2: Ejemplo de configuración de tarea Marathon

Finalmente, para coordinar la asignación de tareas de procesamiento a los nodos se utiliza Zookeeper. Zookeeper permite coordinar tareas de forma distribuida mediante una interfaz de aplicación (API) que permite guardar y recuperar valores de configuración de forma confiable. En particular en la investigación desarrollada en la tesis no fue necesario instalar un servidor dedicado de Zookeeper dado que Mesos incluye un cluster Zookeeper que utiliza para tareas de sincronización internas. El cluster interno de Zookeeper fue utilizado para el intercambio de valores de configuración entre los nodos. Para encapsular el acceso al cluster Zookeeper se utilizó la herramienta zkcli (disponible en el paquete de instalación de Zookeeper) que provee un conjunto mínimo de operaciones para la carga y la descarga de configuración. Es necesario disponer de una interfaz que maneje los diferentes nodos del cluster para mantener los detalles de la topología interna transparentes al código del usuario.

```

while true; do
    sincronizar con Zookeeper para obtener una ruta a una imagen
    obtener el archivo con la imagen a partir de la ruta
    {{ejecutar rutina IRAF sobre el archivo}}
    escribir resultados en salida del sistema
done

```

Código 5.3: Seudocódigo de script de inicialización de nodo

En la Figura 5.1 se presenta la arquitectura básica de funcionamiento para la ejecución del análisis de imágenes astronómicas utilizando la arquitectura Mesos-Marathon. Los resultados de evaluación de performance y eficiencia computacional se describen en el Capítulo 6. El flujo de ejecución involucra los siguientes pasos:

1. Marathon instancia la cantidad de nodos que hayan sido especificados mediante una configuración de tarea (utilizando el procedimiento cuyo seudocódigo se presenta en el Código 5.2)

2. Cada nodo (mediante el contenedor Docker que lo define) obtiene una imagen a procesar de Zookeeper
3. Cada nodo descarga la imagen desde el repositorio de imágenes y realiza el procesamiento
4. Cada nodo reporta los resultados y vuelve a iniciar el ciclo

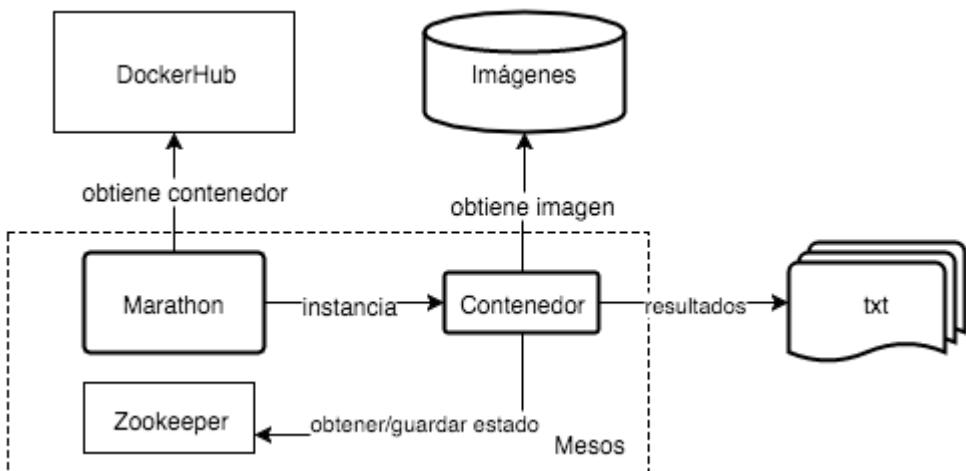


Figura 5.1: Arquitectura de la solución planteada con Docker, Mesos y Marathon

5.1.2. Mejoras en el algoritmo de planificación

En la arquitectura que se diseñó para ejecutar el análisis de varias imágenes simultáneamente, con paralelismo a nivel de datos, en primera instancia no se consideró el ordenamiento de las tareas. La asignación por defecto que ejecuta Mesos, y para la que se obtuvieron resultados iniciales reportados en el Capítulo 6, no tiene implícito ningún concepto de minimización de tiempos medios de ejecución ni maximización de rendimiento.

Para la comparación de algoritmos de planificación de tareas se utilizaron las métricas de *flowtime* y *rendimiento*. En general, se define *flowtime* como la suma de los tiempos de finalización de todas las tareas, y el *mean flowtime* es el promedio de este valor, considerando el número de tareas ejecutadas. El rendimiento es la cantidad de tareas finalizadas por unidad de tiempo.

Considerando la definición de *flowtime* y de rendimiento, se propusieron dos modificaciones en el algoritmo de asignación de imágenes a procesadores de manera de minimizar el *flowtime* y maximizar el rendimiento, dando como resultado el trabajo disponible en [Schnyder et al. \(2017\)](#).

El problema de planificación de tareas aplicado al análisis de imágenes se puede formalizar como:

- Un conjunto de procesadores $P = \{p_1, \dots, p_m\}$
- Un conjunto de tareas (imágenes a procesar) $T = \{t_1, \dots, t_n\}$
- Una función de velocidad $s : P \rightarrow N^+$, donde $s(p_j)$ expresa la capacidad computacional del procesador p_j .
- Una función de peso $w : T \rightarrow N^+$, donde $w(t_i)$ expresa el costo computacional necesario para procesar la tarea t_i .

El objetivo del problema de planificación es encontrar una función $f : T \rightarrow P$ que asigne tareas a los procesadores minimizando w_i/s_j donde $f(t_i) = p_j$. De acuerdo a la clasificación establecida por [Graham *et al.* \(1979\)](#), el problema puede ser clasificado como de tipo $P \parallel C_{max}$.

Algoritmo LPT-CRD

En el algoritmo LPT (del inglés, Longest Processing Time) se ordenan las tareas por mayor tiempo de procesamiento. En el caso del análisis de imágenes astronómicas el tiempo de procesamiento es estimado en base al tamaño de las imágenes. Dado que el código a ejecutar es igual para todas las imágenes, es razonable considerar que mientras más píxeles tenga el archivo, más tiempo se tardará el algoritmo en recorrer la totalidad de la imagen, y por lo tanto más tiempo demorará la tarea. Como resultado de la utilización del tamaño de las imágenes como tiempo estimado de duración para cada tarea se obtiene el algoritmo LPT-CRD que aplica el principio de ordenamiento de LPT al análisis de rayos cósmicos en imágenes, utilizando Docker (el nombre CRD deriva del inglés, Cosmic Ray on Docker). Como resultado de la aplicación del algoritmo, las imágenes más grandes son asignadas con prioridad a los procesadores más rápidos. Un esquema del algoritmo LPT-CRD se muestra en el Algoritmo 2.

El esquema para ordenar CPUs y tareas en el algoritmo LPT-CRD se presenta gráficamente en el diagrama de la Figura 5.2. Los procesadores se ordenan en forma ascendente de acuerdo a su capacidad de procesamiento, y las imágenes de ordenan de forma ascendente de acuerdo a su tamaño.

Algoritmo 2: Algoritmo LPT-CRD para $P \parallel C_{max}$.

```

begin
    Ordenar tareas  $\{t_1, \dots, t_n\}$ , de manera que  $w(t_1) \geq \dots \geq w(t_n)$ 
    for  $i \leftarrow 1$  to  $m$  do
         $c_i := 0$ 
        /* los procesadores  $p_i$  se asumen desocupados desde el
           momento  $c_i=0$  */
    end
     $j := 1$ 
    repeat
         $c_k := \min \{c_i\}$ 
        Asignar tarea  $t_j$  al procesador  $p_k$  at time  $c_k$ 
        /* la primer tarea no asignada de la lista es asignada
           al primer procesador disponible */
         $c_k := c_k + w(t_j)$ 
         $j := j + 1$ 
    until  $j = n$ ; /* todas las tareas han sido asignadas */


---


end

```

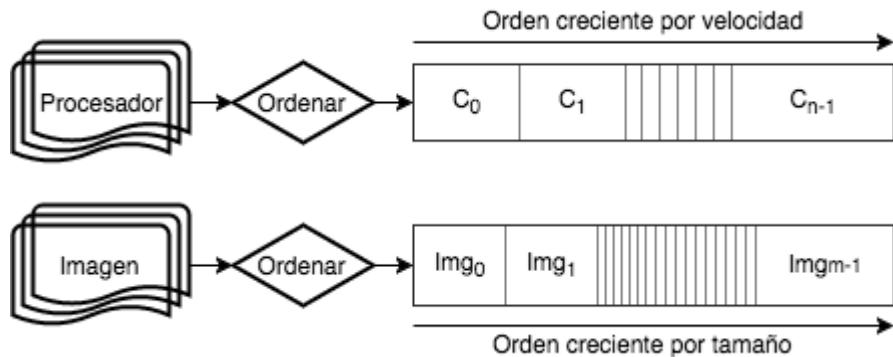


Figura 5.2: LPT-CRD: Distribución de imágenes y CPUs en base a criterios de tamaño y velocidad, respectivamente

Algoritmo CCRD

El segundo enfoque para minimizar el tiempo total de procesamiento de las imágenes propone maximizar el rendimiento. Con tal fin se diseñó una estrategia de asignación simultánea de tareas más largas a los procesadores más rápidos, y de tareas más cortas a los procesadores más lentos. Para establecer ambos tipos de procesadores, se modificó la definición de las tareas a ejecutar en Marathon para simular diferentes velocidades. Como corolario, se establece que los procesadores resultantes son *uniformes*. Por definición, los procesadores

son *uniformes* si cada tarea que ejecuta en un procesador con capacidad de computación s , para t unidades de tiempo, completa $s \times t$ unidades de ejecución.

El algoritmo resultante, denominado CCRD (del inglés, Combined Cosmic Ray on Docker), se presenta en Algoritmo 3. El problema de maximizar el rendimiento también puede verse como un problema de minimización del mean flowtime, por lo que puede clasificarse como de tipo $Q \parallel \sum C_j$ en la clasificación de Graham. Considerando un sistema com m procesadores, y de acuerdo a Horowitz y Sahni (1976), es posible implementar un algoritmo solución cuya complejidad sea $O(n \log mn)$.

Algoritmo 3: Algoritmo CCRD para $Q \parallel \sum C_j$.

```

begin
    Ordenar tareas  $\{t_1, \dots, t_n\}$ , de manera que  $w(t_1) \geq \dots \geq w(t_n)$ 
    for  $i \leftarrow 1$  to  $m$  do
         $c_i := 0$ 
        /* los procesadores  $q_i$  se asumen desocupados desde el
           momento  $c_i=0$  */ 
    end
     $j := 1$ 
     $l := n$ 
    repeat
         $c_k := \min \{c_i\}$ 
        if  $q_k$  es un procesador rápido then
            | Asignar tarea  $t_j$  al procesador  $q_k$  en el momento  $c_k$ 
            |  $j := j + 1$ 
        else
            | Asignar tarea  $t_{n-j}$  al procesador  $q_k$  en el momento  $c_k$ 
            |  $l := l - 1$ 
        end
        /* la primer o la última tarea no asignada de la lista
           es asignada al primer procesador libre, dependiendo
           del tipo */ 
         $c_k := c_k + w(t_j)$ 
    until  $l = j$ ; /* todas las tareas han sido asignadas */ 
end

```

En CCRD, los procesadores y las imágenes son ordenados en orden creciente por capacidad computacional y tamaño, respectivamente. Cada vez que el algoritmo CCRD encuentra un nodo sin asignación le asigna una imagen

disponible. Si el nodo corresponde a un procesador rápido, el algoritmo elige la imagen disponible más grande. Por otro lado, si el procesador es de tipo lento, el algoritmo selecciona la imagen más pequeña. Dado que las imágenes están ordenadas por tamaño, esta asignación se realiza en tiempo constante ($O(1)$).

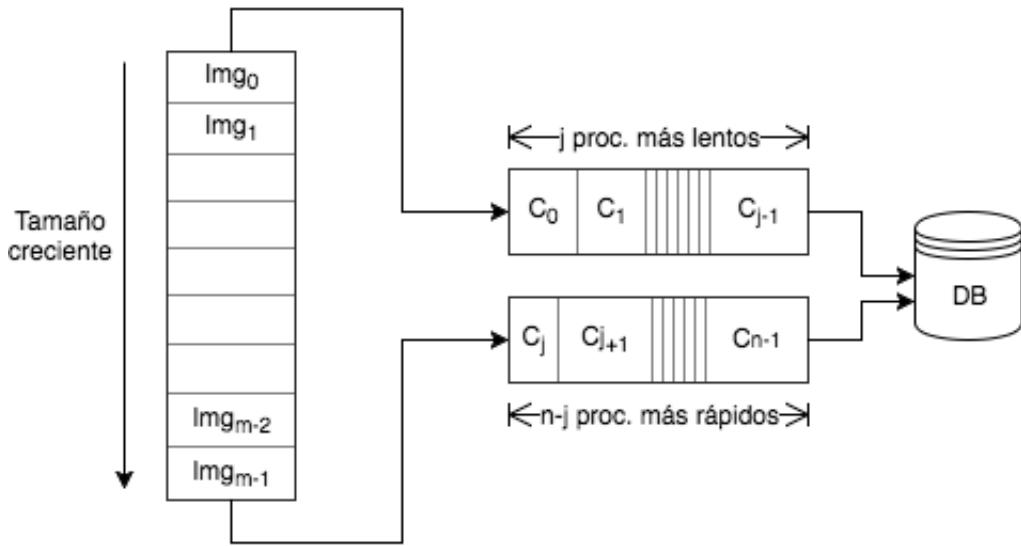


Figura 5.3: CCRD: Esquema de asignación que permite minimizar el mean flowtime

Como se muestra en la Figura 5.3, las imágenes y los procesadores son ordenados de acuerdo a los criterios ya establecidos. Para la diferenciación entre un nodo “lento” y uno “rápido” se utilizó la siguiente lógica: si la configuración de la tarea de Marathon especifica el uso del 100 % del procesador disponible esto define al nodo como de tipo “rápido” y en caso contrario (cualquier valor por debajo del 100 %) como de tipo “lento”. Para probar la arquitectura se utilizó un solo tipo de nodo lento, donde el uso efectivo de procesador equivale al 60 %. Esta clasificación de nodos en “rápidos” y “lentos” es extensible a t tipos, donde cada tipo debe definir un uso del procesador en la escala de 0 % a 100 %.

Las configuración de un nodo lento se muestra en la Figura 5.4 y la configuración de un nodo rápido en la Figura 5.5. Además de la capacidad de procesamiento, se deben incluir otros parámetros como la memoria y la cantidad de nodos a instanciar (para la investigación desarrollada en el marco de esta tesis se utilizaron 5 nodos de cada tipo). Finalmente, se debe proveer a ambos tipos de nodos una cantidad de memoria suficiente para manipular

los diferentes tipos de imágenes sin problemas (como ser excepciones en tiempo de ejecución de los paquetes científicos o disponibilidad de memoria para almacenar los resultados temporales del análisis).

```
{  
  'id': 'slow_worker',  
  'cpus': 0.6,           // porcentaje de CPU reservado  
  'mem': 512.0,          // RAM a ser reservada  
  'instances': 5        // cantidad de nodos a instanciar  
}
```

Figura 5.4: Especificación de una tarea Marathon correspondiente a nodos lentos (utilizando 60 % de la CPU)

```
{  
  'id': 'fast_worker',  
  'cpus': 1.0,           // porcentaje de CPU reservado  
  'mem': 512.0,          // RAM a ser reservada  
  'instances': 5        // cantidad de nodos a instanciar  
}
```

Figura 5.5: Especificación de una tarea Marathon correspondiente a nodos rápidos (utilizando 100 % de la CPU)

Como producto de la investigación de la escalabilidad de la solución utilizando Marathon y Mesos se diseñó un contenedor Docker capaz de ejecutar el análisis de las imágenes y obtener el impacto de los rayos cósmicos. Este contenedor quedó disponible en Dockerhub y puede ser utilizado en otros modelos de programación como ser MapReduce. En particular, se derivó el análisis de escalabilidad y búsqueda de optimización de tiempos de procesamiento como una propuesta de tesis de grado ([Centurión, 2017](#)). Los resultados de la investigación utilizando MapReduce pueden ser comparados en el futuro para tomar este u otro camino al momento de analizar imágenes en formato IRAF procedentes del HST u otro telescopio. En este trabajo se comparan las dos opciones de Mesos/Marathon y Azure Batch.

5.2. Arquitectura utilizando Microsoft Azure

Una vez comprendidos los diferentes aspectos del problema, desde el manejo de los datos hasta la distribución de ejecutables, ciertas características de la solución final quedaron definidas. Por ejemplo, las imágenes deben estar disponibles en un storage replicado y altamente disponible. Se deben evitar cuellos de botella (como se comentó en la sección 5.1) para no retrasar la ejecución del algoritmo por demoras en la sincronización de tareas o en el acceso a los datos. Tampoco puede demorarse la instanciación de nodos si hay recursos de hardware disponibles. Finalmente, los resultados deben quedar disponibles en forma de tablas particionadas para permitir ejecutar consultas filtrando por fecha o por instrumento.

La arquitectura debe comprender entonces cuatro aspectos principales, que pueden asociarse a elementos específicos en la plataforma de Microsoft Azure:

- Almacenamiento de imágenes replicado, mapeado a Azure Storage Blobs
- Scheduling ordenado y no bloqueante, mapeado a Azure Batch
- Nodos virtuales en cantidad arbitraria, mapeado a Azure Virtual Machines
- Resultados particionados y replicados, mapeado a Azure Storage Tables

La Figura. 5.6 presenta un diagrama de interacción de los componentes de Microsoft Batch. La aplicación cliente transfiere los scripts de inicialización y los archivos binarios al servicio de almacenamiento. Luego de transferir los archivos, la aplicación cliente inicia la ejecución de tareas de procesamiento a través de servidores virtuales. La salida de estas tareas de procesamiento también es almacenada en el servicio de almacenamiento.

Un problema a resolver, previamente comenzar la implementación de la solución final, es la elección del lenguaje a utilizar para el código fuente de la aplicación. Como se presentó en la Sección 2.4, el código escrito en Common Language no es adecuado para la distribución de los binarios a los nodos de cómputo, por esta razón el código de la aplicación fue escrito en Python. Se optó por Python debido a que es el lenguaje utilizado en las herramientas astronómicas que construye el departamento de software de JPL y es un lenguaje muy extendido entre los investigadores. Utilizar Python para el código fuente desarrollado en esta tesis presenta dos ventajas: el código resulta de fácil entendimiento y permite además un posible mejoramiento o extensión y el código

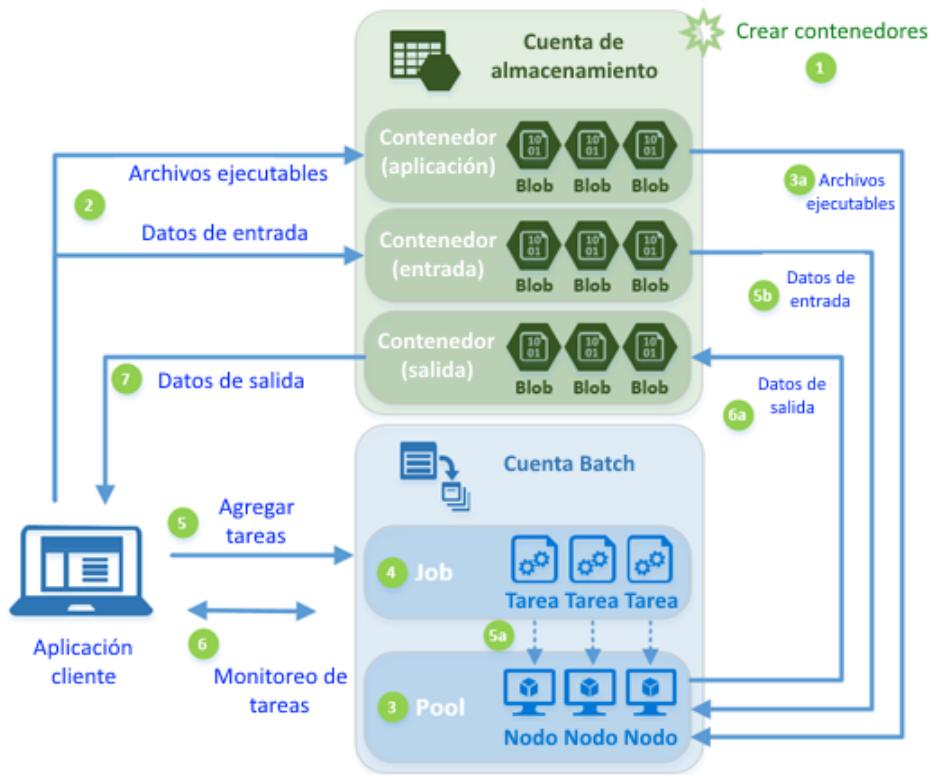


Figura 5.6: Esquema de funcionamiento de Azure Batch [imagen de dominio público, tomada de [Myers et al. \(2017\)](#) y traducida al español]

tiene una la compatibilidad inmediata con toda la suite de herramientas de análisis de imágenes IRAF.

El sistema desarrollado tiene dos responsabilidades principales: en primera instancia procesar la imagen que le sea asignada para obtener los rayos cósmicos presentes; y a continuación calcular con exactitud el posicionamiento del instrumento responsable de tomar la imagen con respecto a coordenadas terrestres. El diagrama de secuencia que representa el sistema construido sobre Azure se muestra en la Imagen 5.7.

Para cada tarea del algoritmo de procesamiento se desarrolló un módulo específico y se utilizaron bibliotecas auxiliares construidas a medida. Para la ejecución de las tareas en Azure Batch no es necesario cumplir con ninguna interfaz o arquitectura en particular (como sí sucede en MapReduce, por ejemplo) pero se debe tener en cuenta que el código tiene que ser mantenible y extensible. Además, Azure Batch provee mecanismos de acceso a los blobs de donde se obtienen las imágenes, mediante la asignación de una llave temporal por el tiempo especificado, que permite el acceso a la información binaria que

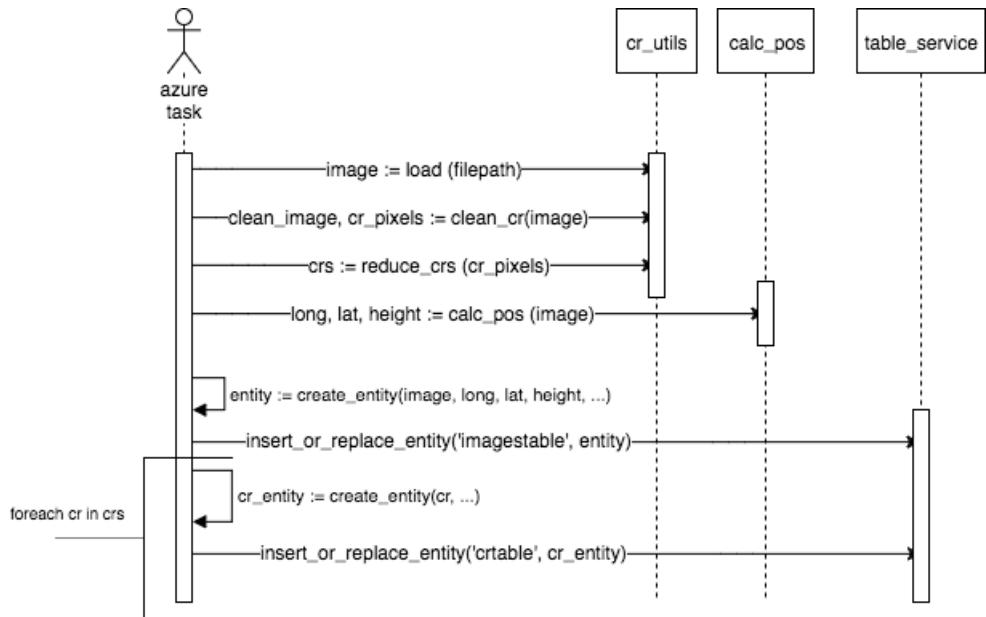


Figura 5.7: Diagrama de secuencia del programa ejecutado en Azure

constituye la imagen y a las bases de datos donde se insertan los resultados. A continuación se presenta y describe una lista de las clases y utilidades desarrolladas para la interacción con Azure y para el procesamiento de las imágenes astronómicas:

- **image.py**

Esta clase modela el formato de las imágenes IRAF y provee utilidades para acceder a los encabezados y a las extensiones directamente. Esta clase encapsula las funciones de las bibliotecas de astropy.iraf.

- **instruments.py**

Esta clase modela los diferentes instrumentos y sus particularidades, por ejemplo el tipo de archivo que contiene los datos y el tipo de archivo que contiene la información de posicionamiento.

- **cosmics.py**

Esta clase implementa el algoritmo lacosmic ([van Dokkum, 2001](#)) para la detección de rayos cósmicos. Es una versión con algunas modificaciones de la implementación original de Malte Tewes, acreditada en el código.

- **crutils.py**

Esta clase provee métodos que permiten cargar una imagen desde un archivo a un modelo propio (image.py), aplicar el algoritmo de detección de rayos cósmicos (cosmics.py) y calcular métricas sobre los rayos cósmicos

detectados.

- calc_pos.py

Esta clase es una traducción de la existente en el código de la prueba de concepto desarrollada en [Tancredi et al. \(2016\)](#). Provee utilidades para calcular la ubicación exacta de la imagen, en términos de la atmósfera terrestre, en base a los parámetros de posicionamiento del instrumento del HST.

- tests

Es un conjunto de clases que validan todas las funcionalidades provistas por la aplicación que se construyó en esta tesis. Se incluyen casos de prueba para todos los instrumentos, desde la carga de los archivos hasta el cálculo de los rayos cósmicos y las coordenadas terrestres correspondientes a cada imagen.

- azure_task.py

Esta clase contiene el código para instanciar el pipeline de detección de rayos cósmicos en cada nodo de Azure Batch. En particular, esta clase contiene las invocaciones a cada método con los parámetros calculados en el paso anterior y el código para persistir los resultados en archivos y en tablas.

- azure_client.py

Esta clase funciona como un orquestador de todo el proceso de análisis de imágenes astronómicas y detección de rayos cósmicos. Gestiona el ciclo de vida del ecosistema, creando el conjunto de instancias y asignando jobs de Azure por cada imagen a ser procesada. La clase también gestiona la planificación de cada job, los timeouts y las expiraciones de los objetos utilizados.

5.3. Manipulación de los datos

Las imágenes capturadas por los instrumentos del HST fueron obtenidas del STScI mediante una solicitud de acceso a los datos con fines de investigación. En general, las imágenes del HST pueden ser obtenidas mediante la plataforma web que habilita el STScI a tal fin, pero la cantidad de datos que se propone procesar en esta tesis hacía inviable esta opción debido al tiempo que sería necesario invertir en la plataforma web para obtener el dataset

completo. El conjunto de darks abarca el período total de funcionamiento del HST, con alternancia de instrumentos debido a los períodos originales de funcionamiento o a interrupciones por fallas o reparaciones, sumando un tamaño total aproximado de 15 TB. Para trabajar con este volumen de datos, los datos se obtuvieron mediante 3 discos. Durante el período previo al acceso a la totalidad de las imágenes se realizaron análisis con los 5 GB disponibles de la prueba de concepto elaborada por los investigadores de la Facultad de Ciencias. Una vez obtenidos los discos con las imágenes, comenzó la tarea de subir los archivos a los servidores de Azure. La operación de subida demandó 10 días aproximadamente. Los discos quedaron conectados a la infraestructura de la Facultad de Ingeniería para otras investigaciones (incluyendo la tesis de grado mencionada en [5.1.2](#)).

Para el almacenamiento de las imágenes se utilizó Azure Blob Storage, una plataforma que permite almacenar grandes volúmenes de datos, con capacidad de replicación altamente disponible. Azure Blob Storage permite disponibilizar los datos almacenados para su uso interno o externo mediante permisos específicos de acceso. Estas características son fundamentales para el tipo de procesamiento que es necesario llevar a cabo, de larga duración por el volumen de datos a procesar y con uso intensivo de la capacidad de cómputo disponible para el procesamiento de cada imagen. Como se muestra en la Figura [5.8](#), la replicación permite evitar cuellos de botella en el acceso a las imágenes de forma concurrente. En primera instancia se asignan las imágenes a los nodos de cómputo para su procesamiento. Cada nodo es responsable de obtener una URL con permisos de acceso mediante un request específico a Azure Batch. Finalmente, cada uno de los nodos establece una conexión propia al blob storage y comienza la descarga a disco para realizar el procesamiento de forma local.

Por otra parte, también fue necesario diseñar una estrategia de almacenamiento de los resultados considerando dos aspectos principales: particionamiento y disponibilidad. Si bien en la propuesta original y en las primeras etapas de esta investigación se utilizó como salida un archivo de texto, esta estrategia no es recomendable para el análisis del conjunto total de los datos. Por ejemplo, si se obtuviera un archivo de salida por cada imagen procesada, sería necesario agregar una nueva etapa de análisis (por ejemplo, un modelo MapReduce) para agregar la información contenida en estos archivos y gene-

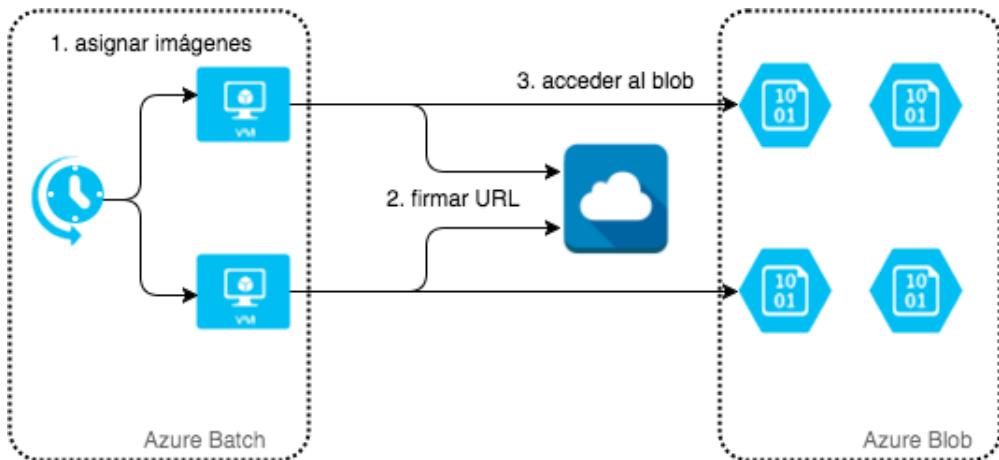


Figura 5.8: Flujo de obtención de acceso a Azure Blob Storage

rar resultados para consultas específicas. La Figura 5.9 presenta una posible arquitectura que permitiría resolver la etapa posterior a la obtención de los resultados del análisis de las imágenes en forma de archivos de salida.

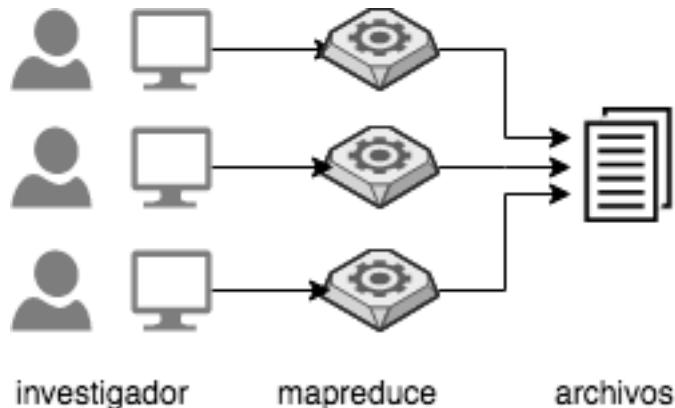


Figura 5.9: Arquitectura para procesar archivos de salida

Como el objetivo de esta tesis es obtener datos para proveer información científica a investigadores e investigaciones en varias áreas, se optó por utilizar un almacenamiento tradicional basado en tablas. Cada imagen analizada es almacenada como una entrada individual en la tabla *images* que contiene la información básica de posicionamiento y la cantidad de rayos cósmicos detectados en esa imagen, mientras que cada rayo cósmico es almacenado como una entrada individual en una segunda tabla *cosmicrays*. Las tablas fueron pensadas bajo un modelo no relacional, donde no se establecen restricciones de clave foránea debido a que no se realizaran joins pero si se establecen claves

de partición y de rango.

images		cosmicrays	
PK	<u>Instrument</u>	PK	<u>image_name</u>
RK	<u>image_name</u>	RK	<u>cr_idx</u>
RK	<u>timestamp</u>	RK	<u>timestamp</u>
cr_count width height <u>exposition_duration</u> longitude latitude <u>< other position parameters ></u>			area orientation <u>< other math attributes ></u>

Figura 5.10: Tablas utilizadas para el almacenamiento de los resultados

La Figura 5.10 muestra la estructura de las tablas donde se almacena la información obtenida de las imágenes. La clave de partición (PK, partition key) de cada tabla determina en qué servidor de Azure Table se persistirá cada entrada de cada tabla. Esta relación entre clave primaria y servidor permite aplicar el principio de localidad de los datos. Este principio establece que cuando se consulta por una entrada es probable que se consulten entradas similares de la misma tabla. Si se asume el principio de localidad, es conveniente almacenar las entradas similares en la misma partición. Además, la clave de fila (RK, row key) es la que permite ordenar las entradas de cada partición por algún criterio en particular (por ejemplo, en orden alfabético) por lo que se pueden realizar búsquedas por nombre de imagen con tiempo de acceso constante. Finalmente, Azure Table incluye un campo de marca de tiempo que utiliza como RK secundario y permite ubicar las entradas por fecha de creación ([Myers, 2017](#)).

En la Figura 5.11 se muestra cómo se distribuyen las entradas en diferentes servidores en base a su clave primaria, donde las entradas con igual clave primaria son almacenadas en el mismo servidor. Los campos elegidos como clave primaria y como clave de fila habilitan a los investigadores a realizar consultas en base al instrumento que deseen, o a un conjunto de imágenes específico con

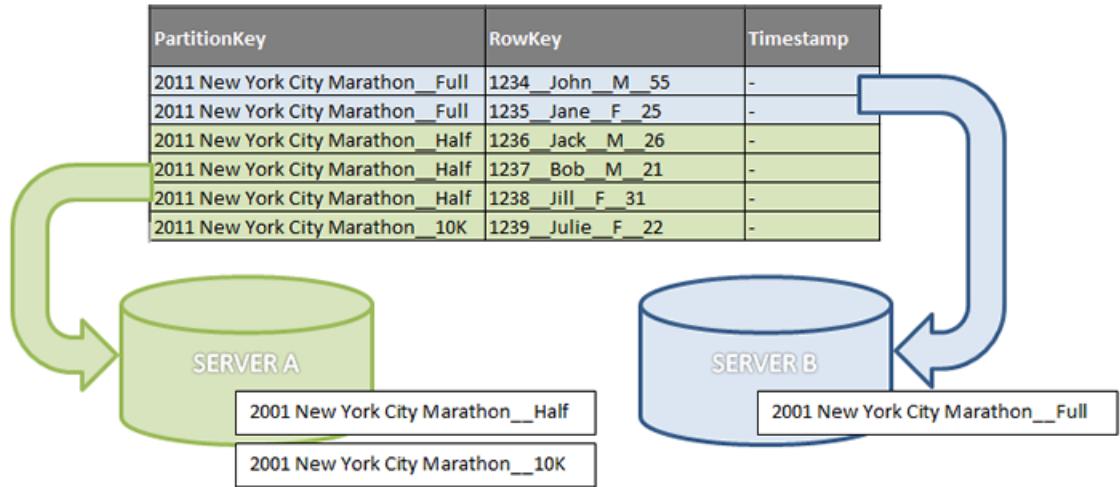


Figura 5.11: Esquema de particionamiento de Azure Tables [imagen de dominio público, tomada de Myers (2017)]

garantías mínimas en el tiempo de acceso. Esta arquitectura de datos, y el subyacente diseño de la aplicación y su interacción con los diferentes componentes de la arquitectura en la nube, establece una base para futuros procesamientos de otros conjuntos de datos (como pueden ser observaciones directas y no solamente archivos de tipo dark). La arquitectura final para analizar la totalidad de las imágenes se presenta en la Figura 5.12.

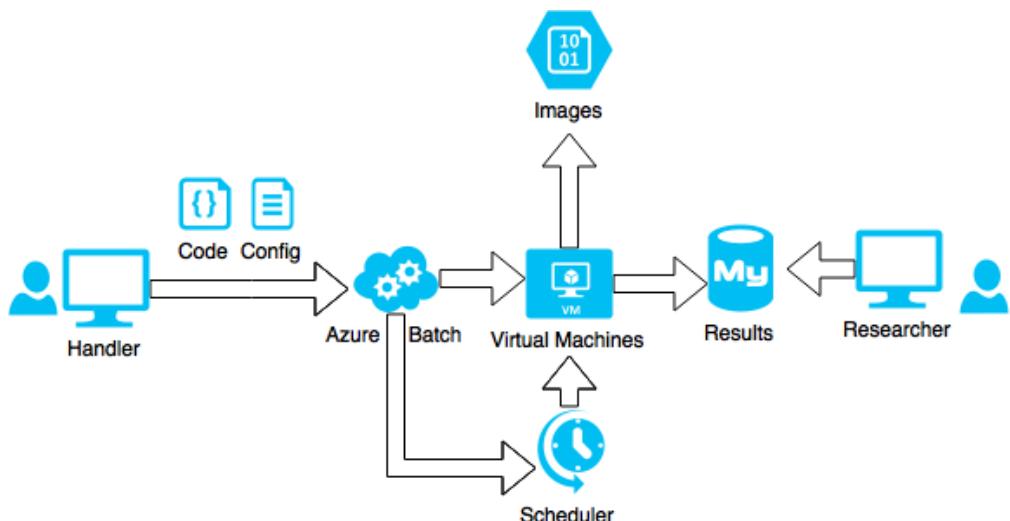


Figura 5.12: Arquitectura de la solución que permite procesar las imágenes astronómicas en Azure Batch y almacenar los resultados en un formato accesible a los investigadores

Capítulo 6

Análisis Experimental

Este capítulo presenta el análisis de eficiencia computacional de la implementación propuesta para el análisis de imágenes astronómicas utilizando Apache Mesos y Microsoft Azure. La primera sección presenta las métricas utilizadas para comparar los tiempos obtenidos en las ejecuciones de ambas implementaciones. La segunda sección presenta el análisis de los tiempos de ejecución del prototipo en Mesos, con detalles de la infraestructura donde se ejecutaron los experimentos y el análisis de los resultados para conjuntos de datos de diferente tamaño. Se incluye además el análisis de las mejoras de performance en el algoritmo de asignación de tareas. En la Sección 5.2 se presenta el análisis de la eficiencia computacional del sistema construido utilizando Microsoft Azure aplicado a la totalidad de los datos. Se analizan los tiempos de ejecución y los inconvenientes que surgieron en la plataforma y en el diseño.

6.1. Métricas

Para la comparación de resultados de eficiencia computacional se utilizan las métricas estándar de *speedup* y *eficiencia*. El speedup se define como la relación entre el tiempo total de procesamiento del algoritmo secuencial (T_1) y el tiempo total de ejecución del algoritmo paralelo (T_N) ejecutado en N de nodos de cómputo. Esta relación se expresa en la Ecuación 6.1. La eficiencia computacional se define como el valor de speedup normalizado por el número de nodos de cómputo utilizados, como se muestra en la Ecuación 6.2

$$S_N = \frac{T_1}{T_N} \quad (6.1) \qquad \qquad E_N = \frac{S_N}{N} \quad (6.2)$$

También se reporta la métrica de *aceleración* para evaluar la mejora relativa, en términos de tiempos de ejecución, obtenida cuando se comparan dos algoritmos, según se define en la Ecuación 6.3.

$$aceleración_{AB} = \frac{tiempo\ de\ ejecución(Algoritmo\ A)}{tiempo\ de\ ejecución(Algoritmo\ B)} \quad (6.3)$$

6.2. Implementación en Apache Mesos

Se realizó un conjunto de experimentos para obtener una comparación precisa de las mejoras obtenidas en los tiempos de ejecución utilizando la arquitectura diseñada sobre Mesos. Los tiempos de ejecución obtenidos por [Deustua y Tancredi \(2013\)](#) no están reportados con el detalle suficiente como para ser utilizadas en las comparaciones de resultados en este trabajo. De todas maneras en el reporte de [Deustua y Tancredi](#) se establece que el tiempo promedio de procesamiento para una imagen de 2048×1024 píxeles es de aproximadamente 1 minuto, mientras que el mejor tiempo obtenido en esta tesis, en el caso de un solo nodo, es de aproximadamente 0,4 minutos. En el caso de la ejecución paralela el mejor tiempo es de 0,05 minutos aproximadamente y corresponde al análisis de 100 imágenes utilizando 20 nodos.

La arquitectura diseñada utilizando Mesos y Marathon se evaluó en una máquina virtual de AWS tipo c4.4xlarge, con un procesador Intel Xeon CPU E5-2680 v2 de 2.80GHz, 16 núcleos, Ubuntu 14.04, 100 GB de disco y 30 GB de memoria.

La Tabla 6.1 presenta los tiempos de ejecución para diferentes conjuntos de imágenes (con 100, 500 y 1000 elementos en cada caso) variando la cantidad de nodos utilizados para el procesamiento. Los resultados muestran que la cantidad óptima de nodos de procesamiento para el análisis de las imágenes astronómicas utilizando Mesos es de 10 nodos. A pesar de que en el conjunto de 100 imágenes se redujo el tiempo en forma proporcional a la cantidad de nodos de procesamiento, en las ejecuciones con 500 y 1000 imágenes los tiempos aumentaron cuando se utilizaron más de 10 nodos.

La Tabla 6.2 presenta los resultados comparativos entre la ejecución secuencial y paralela del código en IRAF CL.

La arquitectura paralela representa una mejora sustancial con respecto a la

# nodos	# imágenes		
	100	500	1000
1	36,80	182,30	434,33
5	10,50	53,18	417,38
10	7,00	150,65	156,87
15	5,93	206,75	376,62
20	5,33	203,75	392,17

Tabla 6.1: Tiempo total de ejecución de la implementación en Mesos para diferentes conjuntos de imágenes, en minutos

# imágenes	tiempo secuencial (min)	tiempo paralelo (min)	speedup	eficiencia
<i>5 nodos</i>				
100	36,80	10,50	3,50	0,70
500	182,30	53,18	3,43	0,69
1000	434,33	417,38	1,04	0,21
<i>10 nodos</i>				
100	36,80	7,00	5,26	0,53
500	182,30	150,65	1,21	0,12
1000	434,33	156,87	2,77	0,28

Tabla 6.2: Análisis de la eficiencia computacional de la implementación sobre Mesos para 5 y 10 nodos y diferentes conjuntos de imágenes

secuencial con algunas consideraciones. Por un lado, el enfoque paralelo o distribuido permite reducir los tiempos de ejecución requeridos para procesar una imagen en hasta $5,26 \times$ (en el caso del procesamiento de 100 imágenes utilizando 10 nodos). Con respecto al speedup y la eficiencia computacional, cuando se utilizan 5 nodos y se procesan datasets de tamaño pequeño, el speedup muestra un comportamiento casi lineal. Con respecto a la eficiencia, cuando se procesan 100 y 500 imágenes se obtiene un valor aproximado a 0,70. Estos resultados, en términos de performance, se pueden considerar como buenos si se toma en cuenta el overhead agregado por el middleware utilizado en la solución propuesta.

Por otra parte, el speedup obtenido muestra un comportamiento sub lineal cuando se utilizan 10 o más nodos. Este comportamiento puede explicarse por una decisión de arquitectura que involucra a Zookeeper. Al utilizarse una sola

instancia de Zookeeper, esta instancia se vuelve un cuello de botella al aumentar la cantidad de nodos de procesamiento. Se puede ver que al aumentar la cantidad de imágenes los tiempos de ejecución también aumentan sustancialmente. Este comportamiento se verificó mediante el análisis de los diferentes componentes de la arquitectura y su interacción en contexto del procesamiento de una imagen. El análisis permitió detectar que la sincronización de configuración que se realiza entre el proceso maestro de Mesos y los esclavos representa demasiada carga para una sola instancia de Zookeeper. Además, los datos referidos a las ubicaciones de las imágenes en los discos compartidos en la red y los tiempos de procesamiento de cada imagen se persisten también en Zookeeper, y esto explica la sobrecarga de transacciones que involucra a la única instancia de Zookeeper.

La Figura 6.1 presenta el comportamiento del speedup con respecto a la cantidad de nodos de procesamiento utilizados. Se puede apreciar el cuello de botella que representa Zookeeper cuando el número de transacciones es alto, especialmente si se utiliza un solo servidor Zookeeper y no un cluster de servidores. Además de volverse un cuello de botella, utilizar un solo servidor de Zookeeper tiene el problema de ser un punto único de fallo, lo que significa que ante cualquier fallo del hardware (o máquina virtual) que ejecuta el servidor se pierde el acceso a la configuración y por ende el sistema deja de funcionar. La única razón por la que se utilizó esta configuración de Zookeeper es que el análisis no se orientó a establecer índices de confiabilidad o resiliencia de Zookeeper sino a validar la plataforma de ejecución de tareas que provee Apache Mesos. En una futura línea de investigación se podría utilizar un cluster de servidores Zookeeper y comparar los tiempos de ejecución para las mismas cantidades de nodos e imágenes que en esta tesis.

La evaluación experimental fue ejecutada utilizando la misma máquina virtual en todos los casos, teniendo en cuenta que se consideran diferentes configuraciones para las tareas de Marathon en los algoritmos propuestos. El uso de CPU máximo fue establecido en el mismo valor (equivalente a 8 procesadores).

La Tabla 6.3 reporta el makespan y el análisis de eficiencia para la implementación de los algoritmos LPT-CRD y CCRD sobre diferentes conjuntos de imágenes en comparación con el algoritmo de asignación de tareas predeterminado de Mesos.

Los resultados reportados en la Tabla 6.3 indican que ambos algoritmos (LPT-CRD y CCRD) tuvieron mejor desempeño que el algoritmo de asigna-

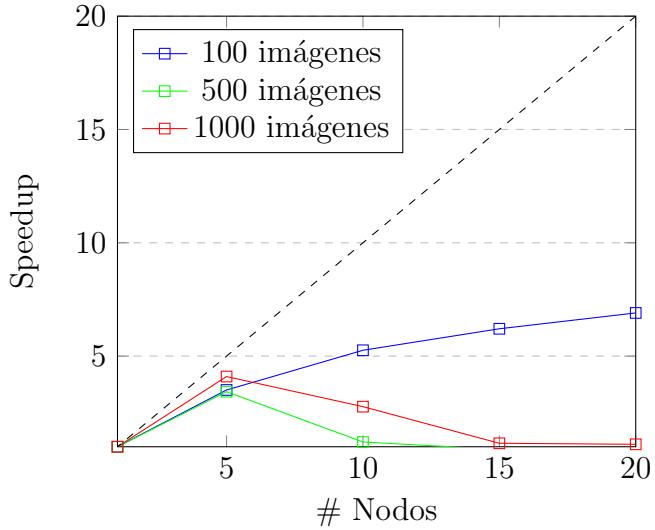


Figura 6.1: Speedup de la implementación en Mesos para diferente cantidad de nodos de ejecución en diferentes conjuntos de imágenes

# imágenes	$\frac{\text{tiempo de ejecución (s)}}{\text{original}} \quad \frac{\text{mejorado}}$	aceleración	speedup	eficiencia
<i>Algoritmo LPT-CRD</i>				
100	346 / 644	0,54	2,83	0,28
500	2382 / 1705	1,40	1,69	0,17
1000	5180 / 3438	1,51	4,17	0,42
<i>Algoritmo CCRD</i>				
100	346 / 533	0,65	3,41	0,34
500	2382 / 1710	1,39	1,69	0,17
1000	5180 / 3326	1,56	4,10	0,43

Tabla 6.3: Resultados de makespan y análisis de eficiencia para la implementación de los algoritmos propuestos sobre diferentes conjuntos de imágenes

ción de tareas implementado por defecto en Mesos. Además, la diferencia de performance crece a medida que aumenta el volumen de datos. Por ejemplo, para el conjunto de 100 imágenes LPT-CRD tiene un speedup de 2,83 y CCRD tiene un speedup de 3,41, pero crece hasta 4,17 y 4,31 para un dataset diez veces más grande. Es notorio que el speedup es menor en el conjunto de 500 imágenes, pero aún así este valor es mejor que el provisto por el algoritmo de asignación de tareas que se implementa por defecto en Mesos. Además, la Tabla 6.3 reporta la aceleración obtenida en las diferentes ejecuciones. En este sentido, los algoritmos propuestos también se comportaron mejor que el algo-

ritmo por defecto, y la aceleración además crece a medida que el tamaño del conjunto de datos crece, lo que indica una relación directa entre la cantidad de nodos de procesamiento y el tamaño del conjunto de imágenes. Las estrategias propuestas permiten ejecutar el procesamiento de imágenes astronómicas de manera más rápida al incrementar el número de nodos de procesamiento.

La Tabla 6.4 reporta el flowtime total considerando los dos algoritmos de asignación de tareas propuestos y el algoritmo implementado por defecto en Mesos, además de las métricas de speedup, aceleración y eficiencia.

# imágenes	tiempo de ejecución (s)		aceleración	speedup	eficiencia
	original	mejorado			
<i>Algoritmo LPT-CRD</i>					
100	186,26	327,44	0,57	2,83	0,28
500	996,83	864,52	1,15	1,69	0,17
1000	2127,65	1739,53	1,22	4,17	0,42
<i>Algoritmo CCRD</i>					
100	186,26	242,67	0,77	3,41	0,34
500	996,83	850,20	1,17	1,69	0,17
1000	2127,65	1708,33	1,25	4,31	0,43

Tabla 6.4: Resultados de flowtime y análisis de eficiencia para los algoritmos LPT-CRD y CCRD sobre diferentes conjuntos de imágenes

Al igual que en la evaluación del makespan, los resultados en la Tabla 6.4 indican que los algoritmos propuestos introdujeron mejoras en los resultados si se considera el flowtime total como métrica a optimizar.

La Figura 6.2 presenta el speedup obtenido con los algoritmos LPT-CRD y CCRD para diferentes conjuntos de imágenes. El gráfico indica que ambos algoritmos tuvieron una performance superior al algoritmo de asignación de tareas que se utiliza por defecto en Mesos para todos los conjuntos de datos. De todas maneras el gráfico muestra que existen diferencias entre LPT-CRD y CCRD y que el speedup aumenta a medida que el volumen de datos se incrementa. La Figura 6.3 presenta la evolución de la aceleración en relación al crecimiento del conjunto de datos para los algoritmos LPT-CRD y CCRD. El gráfico indica un crecimiento sostenido de la aceleración con relación al volumen de datos. Como línea de trabajo futuro se sugiere ejecutar una mayor cantidad de experimentos para generalizar el comportamiento observado en la métrica de aceleración para conjuntos de tamaño superior.

Como conclusión, las mejoras en los algoritmos de asignación de tareas son convenientes para el conjunto total de imágenes del HST. Ambos algoritmos (LPT-CRD y CCRD) introducen una mejora en la velocidad de ejecución total de las tareas de procesamiento obtenida con respecto a la ejecución de tareas utilizando la planificación por defecto implementada en Mesos. El procesamiento del dataset completo de imágenes utilizando Mesos puede acelerarse de forma significativa si se utiliza el algoritmo CCRD, dependiendo del comportamiento del speedup para el volumen del conjunto de datos.

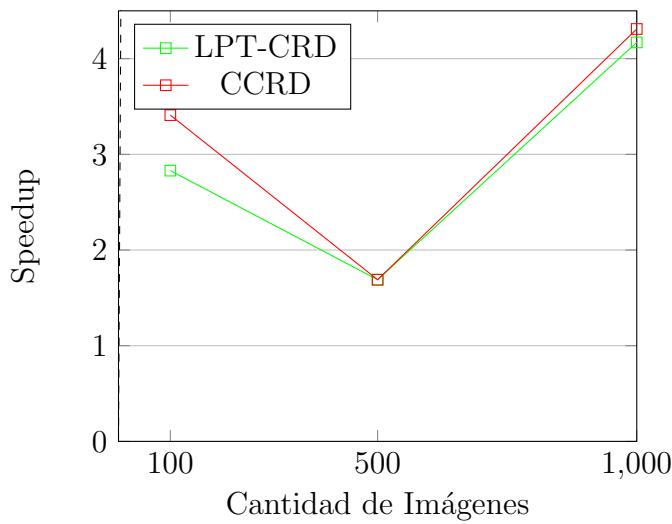


Figura 6.2: Speedup obtenido con los algoritmos LPT-CRD y CCRD para diferentes conjuntos de imágenes

6.3. Microsoft Azure

Para la evaluación del desempeño computacional de la implementación para el procesamiento de imágenes astronómicas desarrollada en Microsoft Azure se utilizó la máxima capacidad asignada de acuerdo a las condiciones de uso establecidas en el patrocinio obtenido con fines de investigación científica ([Roth et al., 2017](#)). La única restricción en términos de capacidad de cómputo es una cuota máxima de 20 núcleos ejecutando simultáneamente siempre y cuando no se excediera el monto total asignado. Dado que las máquinas virtuales utilizadas para el procesamiento eran de doble núcleo, el total de máquinas utilizadas para el procesamiento fue de 10. Cada máquina tuvo instalado Ubuntu 14.04, el procesador fue de tipo Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz y la

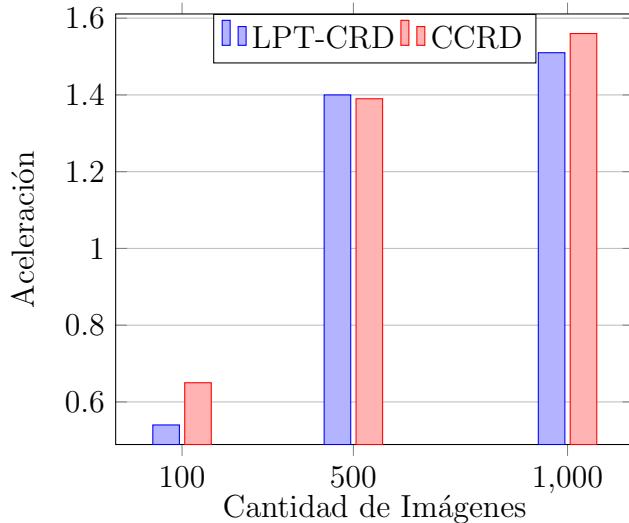


Figura 6.3: Aceleración obtenida con los algoritmos LPT-CRD y CCRD para diferentes conjuntos de imágenes

memoria RAM asignada por nodos fue de 3 GB. Como parte de la inicialización de las tareas de instalaron los paquetes necesarios para la ejecución (por ejemplo python3 y astropy, entre otros). Los tiempos de ejecución de tareas se tomaron directamente de estadísticas reportadas por el portal de Azure, en tanto que las métricas utilizadas para la comparación con la implementación utilizando Mesos fueron calculadas en cada caso.

La Tabla 6.5 presenta las tareas completadas, las tareas por minuto y el tiempo medio de ejecución por tarea para diferentes períodos de tiempo. De acuerdo a los resultados presentados, se observan diferencias de performance cuando varía el período de tiempo considerado. Estas variaciones responden seguramente a la disponibilidad de recursos característica de un proveedor de infraestructura y servicios como Azure. De todas maneras, se aprecia una mejora en el rendimiento total a medida que el tiempo considerado se hace mayor, con efecto directo en la disminución de tiempo total de procesamiento. En un lapso de 5 días se obtuvo un promedio de 15,12 tareas ejecutadas por minuto.

La Tabla 6.6 compara los tiempos de ejecución de la implementación desarrollada en Azure con los tiempos de ejecución de la implementación desarrollada en Mesos. Debido a que la consulta de estadísticas de uso en el portal de Azure se obtiene por período de tiempo, se extrapolaron los resultados para poder compararlos con los resultados de Mesos que figuran por volumen

<i>período (min)</i>	<i>tareas completadas</i>	<i>tareas por minuto</i>	<i>tiempo medio por tarea (min)</i>
60	772	12,67	0,08
1440	18 540	12,88	0,08
7200	82 960	11,52	0,09
14400	217 690	15,12	0,07

Tabla 6.5: Métricas de ejecución de tareas en Azure, por unidad de tiempo

de datos. La comparación se realiza entre la ejecución utilizando el algoritmo CCRD (seleccionada como la mejor ejecución en Mesos) y el tiempo medio de finalización de tarea para el período de 14400 minutos (correspondiente al mejor escenario en Azure).

Los resultados indican que la ejecución en la plataforma de Azure mejoró notablemente los tiempos de las ejecuciones utilizando Apache Mesos en máquinas físicas. Se obtuvo un speedup de 6,57 si se considera el conjunto de 1000 imágenes, lo que significa una eficiencia de 0,66 (considerando que se utilizaron 10 nodos para el procesamiento). Para los conjuntos de 100 y 500 imágenes se obtuvieron speedups de 5,57 y 5,51, respectivamente. Si bien la comparación entre arquitecturas no es justa desde el punto de vista del hardware, se puede destacar que los resultados obtenidos son excelentes en términos de performance y la escalabilidad de Azure permite asignar los recursos necesarios para procesar las imágenes en tiempos muy inferiores a los que serían necesarios en la infraestructura disponible para las ejecuciones con Mesos.

# imágenes	<i>tiempo de ejecución (s)</i>		aceleración	speedup	eficiencia
	<i>CCRD</i>	<i>Azure</i>			
100	533	396,89	1,34	5,57	0,56
500	1710	1984,47	0,86	5,51	0,55
1000	3326	3968,95	0,84	6,57	0,66

Tabla 6.6: Resultados de makespan y análisis de eficiencia para la ejecución en Azure comparada con CCRD, sobre diferentes conjuntos de imágenes

La Tabla 6.7 presenta los tiempos medios de procesamiento obtenidos para cada instrumento, así como el promedio de rayos cósmicos encontrados. Se observa una relación directa entre el tamaño de la imagen y el tiempo que demanda su análisis, y una relación directa entre el tamaño de la imagen y la cantidad de rayos cósmicos encontrados en las imágenes procesadas. De

todas maneras, las imágenes de la cámara ACS muestran una mayor cantidad media de impactos que las de la cámara WFC3, a pesar de tomar imágenes de tamaño similar. Las razones de esta diferencia deben ser analizadas en investigaciones llevadas a cabo por especialistas en el estudio de la radiación cósmica, utilizando los resultados obtenidos en esta tesis.

<i>instrumento</i>	<i>tamaño de imagen (píxeles)</i>	<i>tiempo medio de procesamiento (s)</i>	<i>promedio de rayos cósmicos encontrados</i>
ACS	2048×4096	252,19	375 283,00
NICMOS	256×256	16,91	2368,52
STIS	1024×1024	23,27	20 638,76
WFC3	2051×4096	108,24	54 422,68

Tabla 6.7: Tiempo medio de procesamiento y cantidad media de rayos cósmicos detectados por instrumento

En las primeras ejecuciones utilizando la implementación en python que se realizaron sobre Azure se obtuvieron tiempos superiores a los reportados en los resultados de esta sección. Luego de un análisis profundo de cada etapa del procesamiento se detectó que las operaciones de inserción en la base de datos explicaban la demora en el tiempo de finalización. En general, en las imágenes procesadas existe registro del impacto de varios miles de rayos cósmicos, lo que deriva en igual cantidad de operaciones de inserción en la tabla de rayos cósmicos por cada imagen. La relación entre tiempo de detección de los rayos en una imagen y el número de operaciones de inserción es de 10 a 1 lo que representa un problema si se quiere analizar el dataset completo de imágenes en el menor tiempo posible. Una primera alternativa para solucionar el problema consistió en insertar todos los impactos de rayos cósmicos al mismo tiempo, es decir en una misma transacción, y minimizar el tiempo de autenticación y validación de los requests (estrategia de procesamiento batch o fuera de línea). Sin embargo no es posible ejecutar una operación de inserción de tipo batch en Azure Table para más de 100 entradas, como especifica Microsoft en la documentación de Azure: "The transaction can include at most 100 entities, and its total payload may be no more than 4 MB in size." ([Myers y Shaham, 2017](#)). De todas maneras, se intentó utilizar esta funcionalidad considerando el tope máximo determinado, sin lograr una reducción significativa en los tiempos totales de procesamiento. El procesamiento de una imagen considerando las operaciones de inserción de cada rayo cósmico puede tomar hasta 30 minutos,

de acuerdo a las ejecuciones realizadas. Como consecuencia, en las ejecuciones que se presentan en este capítulo se ejecutó el código sin incluir las operaciones de inserción en la tabla auxiliar para guardar la información de cada rayo cósmico, aunque si se preserva la cantidad de rayos cósmicos disponibles en cada imagen como parte de la entrada que se almacena en la tabla principal de imágenes. Esta decisión no influye en la comparación de tiempos de ejecución con las ejecuciones en Apache Mesos dado que la determinación de preservar la información de cada rayo cósmico se tomó como parte de la arquitectura final y no está presente ni en el trabajo que origina esta tesis ni en las pruebas de concepto realizadas durante la investigación previa.

Capítulo 7

Conclusiones y Trabajo Futuro

Este capítulo presenta las conclusiones de esta tesis respecto a la aplicación de técnicas de cloud computing y computación de alto desempeño al análisis de imágenes astronómicas para la detección de rayos cósmicos. También se detallan brevemente las diferentes líneas de investigación que quedan planteadas de cara al futuro.

7.1. Conclusiones

Esta tesis abordó el problema de procesar grandes volúmenes de datos de imágenes astronómicas para obtener información científica relevante para investigaciones astronómicas y en otras áreas. Los principales desafíos constituyeron en: la obtención de los datos; la disponibilización de las imágenes para el acceso de nodos de cómputo; la interpretación y el análisis de código existente para el procesamiento de imágenes astronómicas utilizando CL, además de su actualización y adaptación para permitir distribuir el análisis del conjunto total de imágenes utilizando técnicas de cloud computing; y finalmente la construcción de un modelo de datos que permita su reutilización por futuros investigadores.

Al considerar el diseño e implementación de un sistema de computación distribuida en la nube, se manejaron varias opciones en base a la disponibilidad de recursos. Se comenzó con un prototipo de lógica encapsulada y distribuida a través de contenedores Docker que pudiera escalar horizontalmente, pero que demostró tener problemas en la construcción de una arquitectura que escalara a nivel de datos. En este sentido, se encontraron limitantes en la utilización de

las tecnologías propuestas dado que no soportaban la intensidad requerida para completar el procesamiento de grandes volúmenes de datos en poco tiempo. El mayor cuello de botella resultó ser la coordinación de tareas con Zookeeper.

En la investigación sobre mejoras en el algoritmo de planificación de cargas de trabajo se propusieron dos modificaciones. El algoritmo LPT-CRD donde se aplica un ordenamiento en base al tamaño de las imágenes y el algoritmo CCRD donde se utilizan dos tipos de procesadores. De acuerdo a las métricas de eficiencia computacional utilizadas en la comparación, ambos algoritmos mejoraron la asignación de tareas por defecto que realiza Mesos. Los resultados indicaron que el makespan total se redujo hasta en un 35 % (con una aceleración del 1.25) utilizando igual cantidad de procesadores rápidos y lentos (5 de cada tipo) para experimentos realizados sobre un conjunto de datos acotado de 1000 imágenes.

Para la programación de la herramienta de análisis de imágenes astronómicas se utilizó Python, por disponer de una amplia variedad de bibliotecas científicas que permitieron resolver diferentes cuestiones numéricas y del algoritmo, pero además porque se adapta bien a diferentes escenarios de ejecución. El soporte para Python que proveen diversos sistemas operativos, así como diversas arquitecturas de procesadores, permite la portabilidad del código desarrollado durante esta investigación. El código se encapsuló en módulos que pueden ser importados desde otros programas Python, pero además se implementó un script que puede ser incluido en el procesamiento por línea de comando estilo bash. Si bien el código desarrollado en esta tesis no ejecuta en múltiples threads, y por ende no ejecuta en forma paralela, se puede instanciar de forma múltiple en esquemas de MapReduce para parallelizar la ejecución a nivel de datos, donde cada instancia del script recibe una imagen, la procesa y devuelve el resultado en la salida estándar. Esta línea de trabajo se está abordando en el grupo de investigación de computación de alta performance de la Facultad de Ingeniería - UdelarR.

Con respecto a la construcción de la infraestructura para el análisis del dataset completo de imágenes, se diseñó una arquitectura escalable para aplicar y trasladar las conclusiones de la fase de investigación a la plataforma Microsoft Azure. Esta plataforma fue la elegida para las ejecuciones dado que se contó con un patrocinio económico de Microsoft con fines académicos. Este patrocinio permitió contar con recursos de cómputo suficientes para elaborar un plan de acción que permitiera llevar a cabo el análisis de la totalidad

de las imágenes en Microsoft Azure. Se buscaron tecnologías que permitieran construir la misma topología que se diseñó en la etapa de investigación, almacenando las imágenes en forma de blobs binarios replicados y disponibles en todo momento. Se utilizó el esquema de jobs y planificadores de cargas de trabajo de Azure Batch para escalar la asignación de tareas a los nodos. Además, se utilizó Azure Table para el almacenamiento de los resultados en forma replicada y con alta disponibilidad. Como resultado se logró procesar la totalidad de las imágenes en 10 días aproximadamente y logrando un speedup de 6,57 en comparación con la ejecución en Mesos.

El modelo de datos resultante refleja las necesidades planteadas al comienzo de la investigación que originó esta tesis. Se provee un mecanismo simplificado de acceso a los datos que permite su utilización de los mismos por parte de investigadores que requieran utilizar la información acerca del impacto de rayos cósmicos para obtener conclusiones relevantes en sus áreas de estudio. Se establecieron pautas para el particionado de las columnas tomando como base el concepto de arquitectura de datos y se estableció un formato que puede ser consultado en una base de datos. En las consultas se puede operar sobre las imágenes individualmente, por conjunto o por instrumento. Además, utilizando el modelo especificado en el diseño de la arquitectura sobre Microsoft Azure, se permite la consulta de las entradas relativas a cada rayo cósmico en particular para obtener sus características, como por ejemplo su orientación, su carga energética, el área de impacto, etc.

La principal contribución de esta tesis es la construcción de un sistema y el diseño de un flujo de datos que permite procesar la totalidad de las imágenes de tipo dark del HST en pocos días. Se proveen además los resultados del procesamiento, bajo las condiciones iniciales estipuladas, y se contribuye con herramientas y conocimiento para poder modificar, adaptar y reutilizar el código generado. A su vez, la información obtenida tras el análisis del conjunto total de imágenes se encuentra disponible para consulta y como suministro de investigaciones relacionadas con el estado y la evolución de la magnetósfera del planeta Tierra y otras investigaciones.

7.2. Trabajo Futuro

Como trabajo futuro se plantea la consideración de algunas alternativas para mejorar los diferentes aspectos tratados en este trabajo. Por ejemplo,

la elección de tecnología estuvo acotada por el acceso a infraestructura de la Universidad y al apoyo en recursos de cómputo que se pudo concretar mediante el patrocinio de Microsoft. Es posible que existan alternativas similares en otras plataformas como Amazon Web Services o Google Cloud Platform. Queda planteada entonces la posibilidad de diseñar arquitecturas similares en estos proveedores de recursos en la nube con el objetivo de comparar la facilidad de implementación y el desempeño computacional del sistema a desarrollar en cada plataforma.

Una alternativa para mejorar el desempeño computacional del sistema desarrollado consiste en aplicar paralelismo a nivel de instrucción en el algoritmo de detección de rayos cósmicos implementado en Python. En el módulo implementado, el recorrido de píxeles de cada imagen se realiza de forma lineal y en un solo hilo de ejecución, pero el código es lo suficientemente flexible para permitir dividir las imágenes en regiones y ejecutar el mismo código para cada región. Incorporar paralelismo multihilos permitiría acelerar el tiempo medio de procesamiento de las imágenes y aprovechar la capacidad de los procesadores con múltiples núcleos. Además, por la forma en que se construyeron los scripts de análisis de las imágenes, se deja planteada la posibilidad de construir esquemas de tipo MapReduce que ejecuten de forma intensiva en un proveedor de infraestructura cloud (como podría ser HDInsight en Azure), donde el núcleo del mapper sea el análisis de la imagen y el reducer agregue la salida de cada imagen por región (de acuerdo a algún criterio de particionamiento).

Como se presentó en el Capítulo 6, existen problemas en la plataforma Azure Table para coordinar las operaciones de inserción de millones de entradas referidas a cada rayo cósmico detectado. La causa principal de este problema radica en la cantidad de inserciones que se ejecutan por cada imagen analizada y en que la operación de tipo batch para la inserción no está lo suficientemente madura en la SDK de Azure disponible para Python. Minimizar el tiempo dedicado a la inserción de entradas en las tablas es clave para completar el ciclo total de procesamiento en un tiempo razonable, por lo que se dejan planteadas otras posibilidades de almacenamiento con diferentes características en el manejo de las operaciones de inserción. Por ejemplo, se podría utilizar una base de datos como Cassandra ([Lakshman y Malik, 2010](#)) cuya característica principal es que permite escribir de manera intensa, dado que internamente mantiene un archivo con los cambios (denominado *commit log*) y solo almacena los cambios en el final de ese archivo. Si se utiliza Cassandra

en una arquitectura de cluster, cada nodo replica la información del commit log al resto de los nodos del cluster con una frecuencia configurable. Se podría entonces reemplazar la capa de almacenamiento de la aplicación desarrollada en esta tesis para que persista el resultado del procesamiento en un cluster de Cassandra y poder mejorar el tiempo total de procesamiento de las imágenes considerando el flujo completo.

Referencias bibliográficas

- Alonso-Calvo, R., Crespo, J., García-Remesal, M., Anguita, A., y Maojo, V. (2010). On distributing load in cloud computing: A real application for very-large image datasets. *Procedia Computer Science*, **1**(1), 2669–2677.
- Astroplan (2017). What are the IERS tables and how do I update them? <http://astroplan.readthedocs.io/en/latest/faq/iers.html>, accedida en julio de 2017.
- Bhasin, H., Khanna, E., y Sudha, S. (2014). Black box testing based on requirement analysis and design specifications. *International Journal of Computer Applications*, **87**(18), 36–40.
- Centurión, G. (2017). *Cloud computing para la detección de rayos cósmicos con datos del telescopio espacial Hubble*. Tesis de grado, Universidad de la República.
- Charbonneau, D., Brown, T., Noyes, R., y Gilliland, R. (2002). Detection of an extrasolar planet atmosphere. *The Astrophysical Journal*, **568**(1), 377–384.
- Christian, E. (2012). Cosmic rays. <https://helios.gsfc.nasa.gov/cosmic.html>, accedida en julio de 2017.
- Dean, J. y Ghemawat, S. (2008). MapReduce. *Communications of the ACM*, **51**(1), 107–113.
- Deustua, S. y Tancredi, G. (2013). An investigation into the feasibility of using cosmic ray events on HST instrument detectors to track changes in geomagnetic field strength. DDRF/JDF Call for Proposals for Fall 2013.
- Dobre, C. y Xhafa, F. (2013). Parallel programming paradigms and frameworks in big data era. *International Journal of Parallel Programming*, **42**(5), 710–738.

- Docker Inc. (2016). Docker - Build, Ship, and Run Any App, Anywhere. <https://www.docker.com/>, accedida en julio de 2017.
- Dressel, L. (2017). *Wide Field Camera 3 Instrument Handbook for Cycle 25 v 9.0*. Space Telescope Science Institute.
- Erl, T., Puttini, R., y Mahmood, Z. (2013). *Cloud Computing: Concepts, Technology & Architecture*. Prentice Hall Press, Upper Saddle River, NJ, USA, 1st edition.
- Feissel, M. y Mignard, F. (1998). The adoption of ICRS on 1 January 1998: meaning and consequences. *Astronomy and Astrophysics*, **331**, L33–L36.
- Foster, I. (1995). *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Graham, R., Lawler, E., Lenstra, J., y Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, **5**, 287–326.
- Grossman, R. (2009). The case for cloud computing. *IT Professional*, **11**(2), 23–27.
- Gunarathne, T., Wu, T., Qiu, J., y Fox, G. (2010). MapReduce in the clouds for science. En *Second International Conference on Cloud Computing Technology and Science*, páginas 565–572.
- Hindman, B., Konwinski, A., Zaharia, M., Ghodsi, A., Joseph, A., Katz, R., Shenker, S., y Stoica, I. (2011). Mesos: A platform for fine-grained resource sharing in the data center. En *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation*, páginas 295–308.
- Hora, J., Patten, B., Fazio, G., y Glaccum, W. (2006). The effects of cosmic rays and solar flares on the IRAC detectors: the first two years of in-flight operation. En D. Dorn y A. Holland, editores, *High Energy, Optical, and Infrared Detectors for Astronomy II*, páginas 6276 – 6276 – 15.
- Horowitz, E. y Sahni, S. (1976). Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM*, **23**(2), 317–327.

- Hunt, P., Konar, M., Junqueira, F. P., y Reed, B. (2010). Zookeeper: Wait-free coordination for internet-scale systems. En *Proceedings of the USENIX Conference on USENIX Annual Technical Conference*, páginas 11–11.
- Intel (2013). Virtualization and cloud computing. <https://www.intel.com/content/dam/www/public/us/en/documents/guides/cloud-computing-virtualization-building-private-iaas-guide.pdf>, accedida en julio de 2017.
- Lakshman, A. y Malik, P. (2010). Cassandra. *ACM SIGOPS Operating Systems Review*, **44**(2), 35.
- Li, J., Humphrey, M., van Ingen, C., Agarwal, D., Jackson, K., y Ryu, Y. (2010). eScience in the cloud: A MODIS satellite data reprojection and reduction pipeline in the windows azure platform. En *IEEE International Symposium on Parallel & Distributed Processing*, páginas 1–10.
- Mackey, G., Sehrish, S., Bent, J., Lopez, J., Habib, S., y Wang, J. (2008). Introducing map-reduce to high end computing. En *3rd Petascale Data Storage Workshop*, páginas 1–6.
- McMaster, M. y Biretta, J. (2008). *Wide Field and Planetary Camera 2 Instrument Handbook v10.0*. Space Telescope Science Institute.
- Mell, P. y Grance, T. (2011). The NIST definition of cloud computing. Technical report, National Institute of Standards and Technology.
- Mesosphere Inc. (2016). Marathon: A cluster-wide init and control system for services in cgroups or Docker containers. <https://mesosphere.github.io/marathon/>, accedida en julio de 2007.
- Mewaldt, R., Cummings, A., y Stone, E. (1994). Anomalous cosmic rays: Interstellar interlopers in the heliosphere and magnetosphere. *Eos, Transactions American Geophysical Union*, **75**(16), 185–193.
- Myers, T. (2017). Designing a Scalable Partitioning Strategy for Azure Table Storage. <https://docs.microsoft.com/en-us/rest/api/storageservices/fileservices/designing-a-scalable-partitioning-strategy-for-azure-table-storage>, accedida en julio de 2017.

Myers, T. y Shahan, R. (2017). Performing entity group transactions. <https://docs.microsoft.com/en-us/rest/api/storageservices/performing-entity-group-transactions#requirements-for-entity-group-transactions>, accedida en julio de 2017.

Myers, T., Macy, M., y Squillace, R. (2017). Get started with the Azure Batch Python client. <https://docs.microsoft.com/en-us/azure/batch/batch-python-tutorial>, accedida en julio de 2017.

Parashar, M., AbdelBaky, M., Rodero, I., y Devarakonda, A. (2013). Cloud paradigms and practices for computational and data-enabled science and engineering. *Computing in Science & Engineering*, **15**(4), 10–18.

Pence, W., Chiappetti, L., Page, C., Shaw, R., y Stobie, E. (2010). Definition of the flexible image transport system (FITS), version 3.0. *Astronomy & Astrophysics*, **524**, A42.

Riley, A. y Biretta, J. (2017). *STIS Instrument Handbook for Cycle 25 v16.0*. Space Telescope Science Institute.

Roth, J., FitzMacken, T., Lian, J., Kemnetz, J., y Squillace, R. (2017). Azure subscription and service limits, quotas, and constraints. <https://docs.microsoft.com/en-us/azure/azure-subscription-service-limits>, accedida en julio de 2017.

Schnyder, G., Nesmachnow, S., Tancredi, G., y Tchernykh, A. (2017). Scheduling algorithms for distributed cosmic ray detection using Apache Mesos. En *Communications in Computer and Information Science*, volumen 697, páginas 359–373. Springer International Publishing.

Smith, E., Keyes, T., Voit, M., Leitherer, C., y Baum, S. (2011). Introduction to the HST Data Handbooks, version 8.0. <http://www.stsci.edu/hst/HST-overview/documents/datahandbook>, accedida en julio de 2017.

Snowden, S. (2011). South Atlantic Anomaly. https://heasarc.gsfc.nasa.gov/docs/rosat/gallery/misc_saad.html, accedida en julio de 2017.

Son, S. (1988). Replicated data management in distributed database systems. *ACM SIGMOD Record*, **17**(4), 62–69.

Space Telescope Science Institute (2017a). Hubble Site. <http://hubblesite.org/>, accedida en julio de 2017.

Space Telescope Science Institute (2017b). Team Hubble: Servicing Missions. http://hubblesite.org/the_telescope/team_hubble/servicing_missions.php, accedida en julio de 2017.

Szabo, C., Sheng, Q., Kroeger, T., Zhang, Y., y Yu, J. (2013). Science in the cloud: Allocation and execution of data-intensive scientific workflows. *Journal of Grid Computing*, **12**(2), 245–264.

Tancredi, G., Cromwell, G., Deustua, S., Gonzalez, G., Nesmachnow, S., y Schnyder, G. (2016). Geophysics using Hubble Space Telescope. Hubble Space Telescope Cycle 24 approved proposal.

The Astropy Collaboration, T. Robitaille, E. Tollerud, E., P. Greenfield, M. Droettboom, E. Bray, T. Aldcroft, M. Davis, A. Ginsburg, Price, A., Kerzendorf, W., Conley, A., Crighton, N., Barbary, K., Muna, D., Ferguson, H., Grollier, F., Parikh, M., Nair, P., Günther, H., Deil, C., Woillez, J., Conseil, S., Kramer, R., Turner, J., Singer, L., Fox, R., Weaver, B., Zabalza, V., Edwards, Z., Azalee, K., Burke, D., Casey, A., Crawford, S., Dencheva, N., Ely, J., Jenness, T., Labrie, K., Lim, P., Pierfederici, F., Pontzen, A., Ptak, A., Refsdal, B., Servillat, M., y Streicher, O. (2013). Astropy: A community python package for astronomy. *Astronomy & Astrophysics*, **558**, A33.

van Dokkum, P. (2001). Cosmic-ray rejection by laplacian edge detection. *Publications of the Astronomical Society of the Pacific*, **113**(789), 1420–1427.

Yeo, C., Buyya, R., Pourreza, H., Eskicioglu, R., Graham, P., y Sommers, F. (2006). Cluster computing: High-performance, high-availability, and high-throughput processing on a network of computers. En *Handbook of Nature-Inspired and Innovative Computing*, páginas 521–551. Kluwer Academic Publishers.

Zimmermann, H. U. (1992). The ROSAT mission. En *Data Analysis in Astronomy IV*, páginas 115–119. Springer US.