UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS

HABILITACIÓN PROFESIONAL EN GEOFÍSICA

MODELACIÓN NUMÉRICA DEL OCÉANO EN NUBE Y SISTEMAS HPC.

Por: Bruno Herrera Sobarzo

Junio 2023

Concepción, Chile

Profesor Guía: Andrés Sepúlveda

Comisión: Diego Narváez - Nicolás Sepúlveda

AGRADECIMIENTOS

Primero quiero partir agradeciendo a mis Padres, Marina y Robinson, siempre ayudándome a crecer y ser mejor persona. A Mi familia, mi hermana Camila, tías/os, primas/os y sobrino, cada uno forma parte de mi vida, siempre me dieron sus consejos y compañía. A todos le agradezco de entregarme amor.
Para mis amigos, que me ayudaron a no rendirme en los momentos mas difíciles, hicieron mi etapa universitaria una muy buena experiencia, quiero agradecerles; Daniel Gonzalez y Aaron Acuña, la mejor trifuerza; Tito Peña, Manuel Suazo, Ignacio Partarrieu, Scarlett Moraga, Catalina Llancaleo, Vicente Valenzuela, Camilo Escalona y Juan Escalona por su cariño.
Agradecer a Monserrat de la Jara, por darme todo su cariño, amor y todo el apoyo que me brindaste en este momento.
Al Departamento de Geofísica. A los profesores y funcionarios, que se ve un esfuerzo constante para brindar siempre lo mejor a los estudiantes. En especial a Carolina Parada por estar en gran parte de mi formación.
Agradezco también a mi Profesor guía, Andrés Sepúlveda, por ayudarme y darme apoyo durante todo este proceso, brindándome oportunidades para enseñar mis conocimientos y también seguir aprendiendo.A la comisión, Diego Narváez y Nicolás Sepúlveda, quienes no tuvieron problemas en resolver dudas o conversar sobre este trabajo.
Finalmente quiero dedicar este logro a mi abuelita Guillermina Rivas, estaré eternamente agradecido por todo tu amor, siempre seras mi razón para seguir adelante.

Agradecer formalmente el financiamiento del proyecto Fondecyt Regular 1211230 del Dr. Héctor Hito Sepúlveda.
Esta investigación/tesis fue parcialmente apoyada por la infraestructura de supercómputo del NLHPC (ECM-02)

Resumen

El desarrollo de la investigación científica en los últimos años ha ido de la mano con el avance de las nuevas tecnologías y también de los recursos informáticos. Esto ha generado un beneficio en la investigación científica y modelos numéricos, estos últimos siendo cada vez más representativos y de mayor resolución. Debido a esto se requiere de un mayor poder computacional, el cual es abordado por la computación de alto rendimiento (HPC),computación en grilla, con super-computadores y clúster, utilizando técnicas de paralelización, optimizando software y programas. Con la mercantilización, necesidad de acceder de estos recursos informáticos e internet, nace el servicio de la Nube, que ofrece plataformas, infraestructura y diferentes herramientas hechas a la medida del usuario, cuyo un modelo de negocio es el pago por uso. Esto elimina la necesidad de actualizar los equipos constantemente y también abre oportunidad de acceder a diferentes recursos informáticos de forma más flexible y escalable. En este trabajo, se exploraron nuevas opciones de desarrollos que pueden ser aplicadas al modelo numérico del océano *Coastal and Regional Ocean COmmunity model* (CROCO) en la nube. Se realizaron pruebas en el proveedor Microsoft Azure, utilizando el servicio de pago por uso para una máquina virtua(MV) de las series Fs_v2 con 4 vCPU. Se lleva a cabo una simulación con características realistas, estimando el tiempo de cálculo, también se realiza un escalamiento y paralelización, utilizando diferentes distribuciones para estimar eficiencia y *Speedup*. Como resultado se detalla el procedimiento para realizar esta implementación y se obtiene una mayor eficiencia para subgrillas distribuciones del dominio de paralelización en dirección meridional para el caso de esta simulación. Se plantean otras alternativas en desarrollo enfocadas en el cómputo de alto rendimiento, experiencias, se realiza estimaciones de costos a simulación a 20 años, se compara con el NLHPC, se platean alternativas HPC para aplicaciones futuras y opciones para generar una disminución en los costes en la nube. Concluyendo que la alternativa aplicada es accesible para usuarios nuevos en la nube pero presenta costes que es necesario comparar con infraestructura local o propia para saber si son óptimos para su elección y el como obtener financiamiento en el caso de necesitar este servicio.

Índice general

Índice de cuadros

Índice de figuras

Capítulo 1

Introducción

1.1. Computación en la ciencia

En las últimas décadas, la ciencia computacional ha tenido un crecimiento de forma exponencial, lo que ha permitido un avance en el desarrollo científico, basado en simulaciones de fenómenos muchos más complejos, todo este avance científico ha sido denominado parte del e-science, que según Bohle (2013):
"Es la aplicación de la tecnología informática a la realización de la investigación científica moderna, incluida la preparación, la experimentación, la recogida de datos, la difusión de los resultados y el almacenamiento y la accesibilidad a largo plazo de todos los materiales generados a través del proceso científico".
Debido a la importancia de esto último, se han mercantilizado los recursos informáticos, produciendo un aumento de datos procedente de instrumentos más accesibles y soluciones en modelaciones numéricas más estables, que han cambiado el paradigma en la ciencia (Vance et al., 2016).Por lo que la e-science es un método científico transformador que ha ayudado la interacción y el compartimiento de datos y conocimientos científicos dentro de la misma comunidad.
Conforme se avanza en la investigación y se generan modelos numéricos más complejos, necesitamos un poder computacional mayor con mayores requerimientos y en algunos casos un almacenamiento para la generación o muestra de datos, sobretodo cuando la máquina propia no tiene el poder computacional necesario(Vance et al., 2016).

1.2. Soluciones para mayor poder computacional

Una de las soluciones más empleadas son los clústers, este tipo de computación se centra en el trabajo de computadoras independientes que juntas realizan tareas en una misma área geográfica conectadas por una red de alta velocidad, aunque estos presentan costes bastante elevados(Li et al., 2016).

Otro ejemplo es la computación en malla (*grid computing*), esta esta tecnología posibilita el desarrollo de la computación distribuida a gran escala. Permitiendo acceder a recursos de computación y datos en todo el mundo, siendo computadores conectadas en una red de área amplia (WAN) y geográficamente dispersas.(Ahmed et al., 2020) Aunque esta presenta algunos problemas, como la latencia,asociados a la distribución geográfica, especialmente si se involucra una transferencia de datos o una comunicación de la ejecución de una tarea que requiera una comunicación con una conexión estable (Vance et al., 2016).

Otra opción que surge para programas o softwares que requieran gran poder computacional es la computación de alto rendimiento o *High Permormance Computing*(HPC), este aprovecha la tecnología y poder de las supercomputadoras o grupo de computadoras para resolver problemas de computación masiva, ayudando a procesar grandes cantidades de datos o procesos más complejos de manera más rápida, en comparación a una computadora estándar, ya que se utilizan todos los recursos o procesadores disponibles para completar muchos trabajos a la vez.

Un enfoque relevante en la computación de alto rendimiento es la resolución de tareas de forma paralela, utilizando múltiples procesadores o nodos para la ejecución. Pero el número de procesadores no es el único factor que influye dentro de los trabajos de HPC, necesitamos discos rápidos, memoria de alta velocidad (RAM), una baja latencia (en el caso de clúster) y también en algunos casos procesamiento de gráficos(GPU)(IBM, 2021).

Con este tipo de soluciones podemos generar un clúster HPC formado por un nodo principal y varios nodos de computación. El nodo es principal encargado de organizar y designar tareas y los demás de ejecutar los trabajos designados. Actualmente, este tipo de clúster son utilizados en diferentes sectores, como el aeoroespacial, las ciencias de la vida, finanzas, energía, etc(Netto et al., 2018).

No todo es positivo con la computación HPC. Uno de los mayores desafíos con los sistemas clúster HPC es que son bastantes costosos, requieren tiempo,

planificación y la necesidad de un conocimiento avanzado de informática, y de no contar con esto mencionado con anterioridad, se debe contratar a un equipo o profesional capacitado para este tarea(Vance et al., 2016).
Además, los clúster presentan problemas complejos en cuanto a las aplicaciones o programas que se quiera desarrollar, esto debido a la configuración de los clústers es específica, y limitada al personalizar el hardware y software. Estas dificultades se pueden resumir en que el clúster esta diseñado para ejecutar un tipo especifico de tarea de computación con los requisitos hardware y software de las aplicaciones o programas, lo que requiere reconfigurar el entorno del sistema, donde no siempre se puede configurar de manera fácil o rápida(Vance et al., 2016).
Otro aspecto a tener en consideración es la vida útil que puede un clúster, algunos estudios como Siuta et al. (2016) estiman una vida útil de aproximadamente de 5 años, aunque esto varía de acuerdo al mantenimiento, diseño, nivel de uso, componentes utilizados y sujeto a la obsolescencia(Li et al., 2016).
Por último, el clúster HPC designa tareas mediante programadores de gestión por *batch shoulders*(lotes) que reciben las solicitudes de los usuarios para ejecutar trabajos, que se ponen en cola. En momentos los grandes clústers incurren en gastos innecesarios cuando no hay una buena gestión, generando un gran desprecio de recursos y una baja calidad de servicio(Netto et al., 2018).

Una alternativa más actual es el cómputo en la nube. La nube surge debido debido al aumento de la mercantilización de recursos y el auge de internet, este objetivo se basa reducir el coste de la informática, aumentar la fiabilidad y aumentar la flexibilidad transformando los ordenadores de algo que compramos y manejamos nosotros mismos a una responsabilidad gestionada por un tercero. Esta solución comparte objetivos y conceptos con la computación en malla, donde podemos señalar que la nube ha evolucionado a partir de esta y la considera como columna vertebral e infraestructura de apoyo. La evolución ha sido el resultado de un cambio de enfoque de una infraestructura que ofrece recursos de almacenamiento y computacionales, a una basada en la economía, ofreciendo recursos más abstractos y escalables(Foster et al., 2008).

1.3. Cómputo en la Nube

La historia del término Cloud o "Nube"surgió en las décadas de 1950 y 1960 por John McCarthy, y en los años 90 se convierte en una realidad gracias a internet. *Salesforce.com* introdujo el concepto de "Software as a Service"(SaaS) en 1999, mientras que Amazon lanzó Amazon Web Services (AWS) en 2002, y Google creó Google Docs en 2006, llevando el concepto de la nube al público en general. En ese mismo año, Amazon lanzó Elastic Computing Cloud (EC2), permitiendo a las empresas alquilar computadoras para ejecutar aplicaciones a través de la web.(Del Vecchio et al., 2015).

El término de cómputo en la nube o los servicios de cómputo de la nube responde a una necesidad de acceso a diferentes recursos informáticos, la virtualización , el recambio de la utilización de los computadores en red y esto hoy en día nos encontramos en nuestro diario vivir. Existe diversas plataformas ofrecen un sinfín de herramientas para la realización trabajos y gestiones para nuestra vida diaria, de los más conocidos y consolidados, Microsoft Azure, Google Cloud y Amazon Web Services(AWS). Con este tipo de servicio podemos acceder a un conjunto compartido de recursos informáticos configurables y virtualizados, como pueden ser redes de servidores, almacenamientos, software y servicios, todos alojados en centros físicos y permitido tener un acceso omnipresente por medio de la red. El beneficio que se puede obtener al tener un acceso de forma más inmediata puede influir directamente en el trabajo o investigación en las comunidades científica, sobretodo si tenemos un gran número de tareas en nuestras máquinas o clúster, que al final del día solo irán agregando a la fila(Aljamal et al., 2018).

Las principales ventajas de estos servicios son el pago solo por el uso de recursos, la flexibilidad a la hora de solicitar recursos (ya sea un aumento o disminución de estos), la eliminación de la necesidad de llevar a cabo tareas de mantenimiento en equipos o cambio de estos, la seguridad de tener recursos fiables, no generación de cola de espera (como suele suceder en clusters HPC), un coste acorde al uso, una respuesta rápida a problemas que necesiten una solución más inmediata, escalabilidad y privación de recursos y una flexibilidad para brindar diferentes entornos para programas o softwares (Aljamal et al., 2018). Pudiendo ser este servicio un buen apoyo al desarrollo de la investigación científica, modelación numérica y utilización de aplicaciones o modelos en los cuales sean más eficientes utilizando computación de alto rendimiento.

La definición de cómputo en la nube, es básicamente almacenar y acceder a los datos y programas de internet en lugar de un disco duro de nuestro ordenador. El término nube es una metáfora que solemos representar a internet como nube. En esta hacemos el uso de hardware y software donde podemos ofrecer servicios (Gmail de Google, por ejemplo)(Rashid and Chaturvedi, 2019).
Este es un modelo que permite el acceso ubicuo y cómodo a la red bajo demanda a un conjunto compartido de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente aprovisionados y liberados con un mínimo esfuerzo de gestión o de interacción con el proveedor de servicios a través de la virtualización(Mell et al., 2011). Contamos con proveedores de los servicios de la nube, que podemos ver como vendedores que proporcionan al cliente recursos y servicios de computación según el modelo de negocio del cliente.
Se tienen diferentes tipos de definiciones de nube, clasificadas de la siguiente forma(Rashid and Chaturvedi, 2019):

- **Nube privada:** Dirigida para trabajar con solo una organización o negocio en específico.
- **Nube pública;** Son dedicada para un fácil acceso y proveen infraestructura y servicios para el publico y cualquier organización.
- **Nube comunitaria:** Servicio e infraestructura dirigido para una organización de un interés similar.
- **Nube híbrida:** Es una combinación entre la nube privada y una pública.

Y se pude clasificar en 3 los servicios que nos ofrece la nube(Wankhede et al., 2020):

- **Software as a Service (SaaS):** Software como un servicio, corresponde a alojar un programa informático en una plataforma en la nube para evitar instalaciones innecesarias. También permite una interfaz de esta aplicación que puede estar dirigida al cliente mediante un navegador web(un ejemplo de esto puede ser Gmail).
- **Platform as a Service (PaaS):** La plataforma como servicios trata de un proveedor de servicios en la nube ofrece, ejecuta y mantiene tanto el

software del sistema (es decir, el sistema operativo), este incluye el diseño desarrollo y alojamiento de aplicaciones.

- **Infrastructure as a Service (IaaS):** La infraestructura como servicio comprende proporcionar un conjunto de recursos informáticos virtualizados como la CPU, la memoria, el sistema operativo y software de aplicación, etc. Osea, máquinas virtuales(MVs) y almacenamiento.

En los últimos años se ha producido una migración drástica de aplicaciones y servicios a la nube(Gill and Buyya, 2019), ya que como mencionamos anteriormente cuenta con diferentes beneficios, externalizando recursos informáticos para el usuario. También debido a la virtualización de los recursos se han podido gestionar entornos HPC en la nube, denominados cloud HPC, que alejado de los entornos HPC tradicionales, los usuarios pueden ajustar rápidamente sus reservas de recursos, mediante un mecanismo conocido como elasticidad, debido al tamaño de las plataformas gestionadas por los grandes proveedores de computo de la nube(Netto et al., 2018).

1.4. Microsoft Azure

Como mencionamos al principio existen diferentes proveedores servicios en la nube, pero para el fin de este trabajo se trabajará con Microsoft Azure, de los servicios más populares actualmente. Esta selección es debido a que en el desarrollo previo a la creación del proyecto de habilitación se trabajó creando instancias y desarrollando las simulaciones mayoritariamente en este entorno. Este servicio tiene un gran soporte, una sección donde podemos generar auto aprendizaje y diferentes servicios que van desde almacenamientos, MVs, clústers Kubernetes, etc. Todos alojados en centros ubicados en diferentes regiones del Mundo.
Dependiendo de la elección de nuestro servicio, podemos realizar nuestro presupuesto aproximado seleccionando diferentes servicios a la vez, esto se puede realizar mediante una calculadora de precios que contiene la página de Microsoft Azure(https://azure.microsoft.com/es-es/pricing/calculator/)

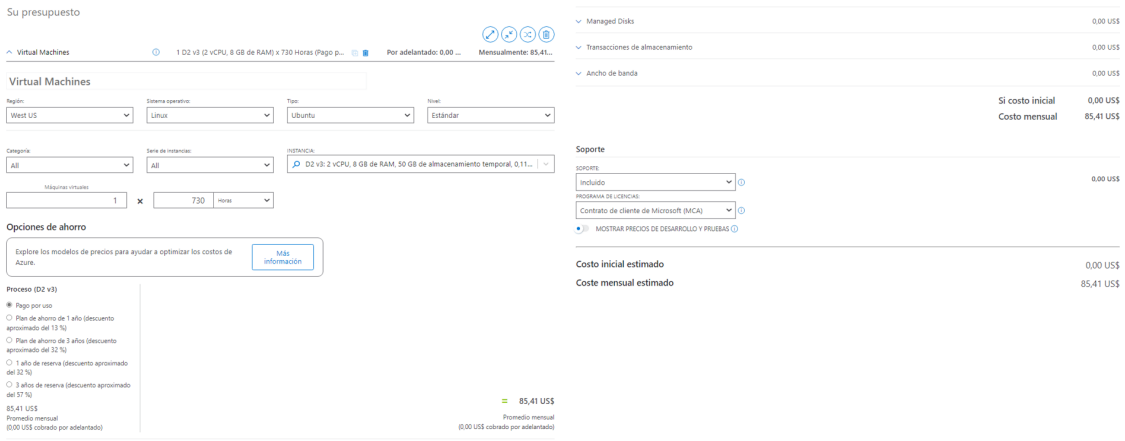


Figura 1.4.1: Ejemplo de presupuesto para una Instancia de máquina virtual en Microsoft Azure.

Por ejemplo, para las máquinas virtuales se considera la región donde será montada nuestra máquina virtual(MV),el sistema operativo, la clasificación de la instancia, tipo, espacio y tipo de disco, si tenemos transacciones de almacenamiento, ancho de banda y también la transferencia de datos de salida, ya que no considera cobro por datos de entrada (hasta la fecha). Ahora también considerar un soporte por parte de Microsoft y tendríamos un valor estimado con un coste inicial y mensual. Estas opciones dependerán del servicio seleccionado o si es más de uno.

1.5. Modelo numérico del océano

En lo que respecta a modelación del océano, existen variados modelos oceánicos que en los que uno puede desarrollar simulaciones de circulación, marea, oleaje, etc. Estos modelos juegan un papel muy importante para ayudarnos a comprender y predecir la dinámica del océano(Jung et al., 2021).En lo que respecta a la comunidad científica nacional, la mayoría de las modelaciones numéricas del océano utilizadas para realizar trabajos de investigaciones es el modelo *Regional Ocean Modeling System*(ROMS).

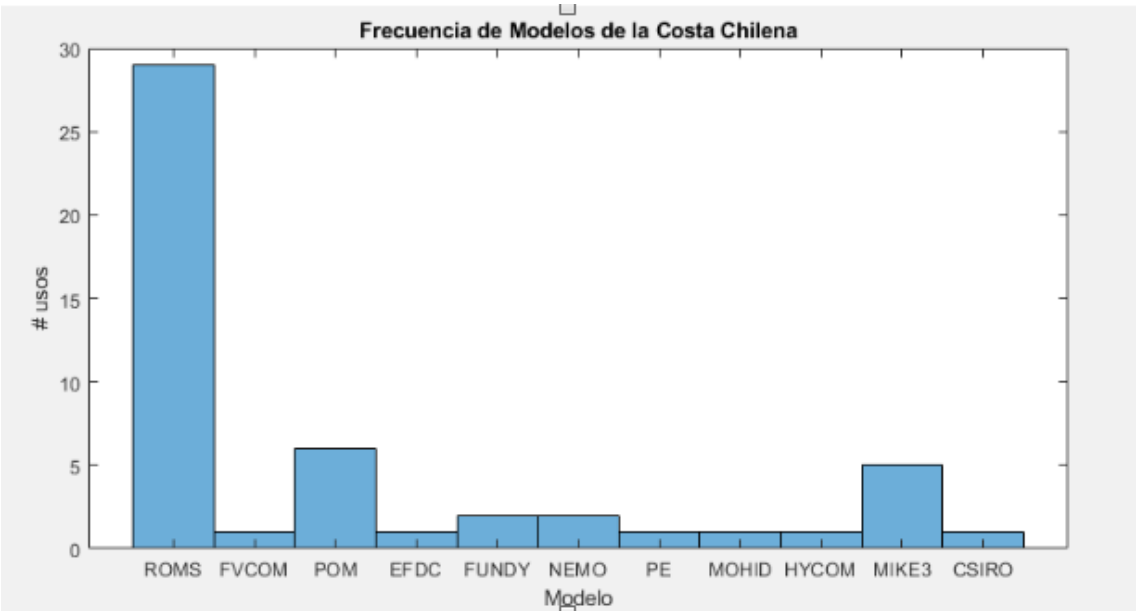


Figura 1.5.1: Frecuencia de uso de modelos del océano en investigaciones por la comunidad científica chilena.(Imagen facilitada por Andrés Sepúlveda)

Por lo que a la hora de seleccionar un modelo numérico del océano para utilizarlo en la nube es necesario entender el contexto nacional de los trabajos realizados, esto con el fin de generar un alineamiento con el desarrollo de investigaciones y generar material de utilidad para el uso de una nueva herramienta como lo es el cómputo de la nube. Con esta idea, se selecciona el modelo numérico *Coastal and Regional Ocean COmmunity model* (CROCO), que está construido en ROMS_AGRIF y el núcleo no hidrostático de SNH(CROCO, 2020), el cual es utilizado en diferentes investigaciones de la comunidad tales como:

- CHONOS: Oceanographic information website for Chilean Patagonia(Reche et al., 2021).
- Subsurface Mesoscale Eddy Generation in the Ocean off Central Chile (Contreras et al., 2019) .
- The impact of spring-neap tidal-stream cycles in tidal energy assessments in the Chilean Inland Sea (Artal et al., 2019).
- A novel approach to assess the hydrodynamic effects of a salmon farm in a Patagonian channel: coupling between regional ocean modeling and high resolution les simulation (Herrera et al., 2018).

El modelo oceánico CROCO(Auclair et al., 2022) es un modelo realista, que está basado en la aproximación de Boussinesq, momento hidrostático y balance de masas, conservación de traza la ecuación de esta para el agua de mar(no lineal) y parametriza procesos de transporte de sub-grilla. Su integración temporal se realiza con una descomposición de los campos 3D en barotrópicos (promediados en profundidad) y baroclínicos (residuales) para facilitar facilitar el cálculo de la fuerza de gradiente de presión(Shchepetkin and McWilliams, 2005).
Además CROCO contiene una *toolbox* llamada CROCO_TOOLS(Penven et al., 2022) que nos permite generar archivos de entrada con herramientas en Matlab para el modelo. El requerimientos de espacio básico para la instalación de CROCO y CROCO_TOOLS es menos de 500 MB, también dependiendo de la configuración regional de este modelo se recomienda un espacio de 18 GB en el disco de almacenamiento de nuestra máquina(Jullien et al., 2022). Otros requerimientos que este necesita son(CROCO, 2023b):

- Un compilador de C
- Librería de Fortran
- Librería de Netcdf
- Librería MPI(en el caso de correr el modelo en paralelo)

Por lo que es posible instalar estos requerimientos fácilmente en una máquina con imagen de Linux. Dependiendo del tipo de simulación que uno quiera realizar, puede variar estos requerimiento, sobretodo para simulaciones más realistas.
El modelo cuenta con diferentes módulos para poder abarcar diferentes ámbitos de la modelación numérica del océano, desde un módulo no hidro-estático, incorporación de mareas, módulos biogeoquímicos, módulo de sedimentos. También el modelo permite realizar anidados online (one-way y two-way) u offline, también nos permite la opción de agregar ríos . Por último, realizar liberación de partículas de manera online y offline. CROCO también cuenta con la opción de realizar acoplamientos con otros modelos, tales como WW3, WRF y Meso-NH(CROCO, 2023a).
Otra característica con la que cuenta CROCO es la capacidad de paralelizar la simulación, lo que permite optimizar el tiempo de cómputo. Con esto podemos estimar el rendimiento de nuestra simulación dependiendo de nuestros procesadores o nodos que tengamos a disponibilidad y de esta misma forma ir escalando con el

fin de optimizar el tiempo de cálculo. Para la realización de esta paralelización, CROCO nos permite su uso tanto en arquitecturas de ordenadores paralelos de memoria compartida o distribuida. Donde podemos asignar múltiples sub-dominios a cada procesador para optimizar el uso de la memoria caché del procesador. Esto permite el escalamiento que mencionamos anteriormente. Tenemos dos alternativas: MPI(memoria distribuida) y OpenMP(memoria compartida), aunque no el uso de ambas opciones al mismo tiempo(paralelización híbrida)(CROCO, 2020).

1.6. Experiencias en la nube

Como el avance del cómputo en la nube es inminente y cada vez más accesible, es necesario poder realizar un análisis y factibilidad de utilizar las herramientas de cómputo de la nube, ya sean MVs, clúster, almacenamiento, etc. Así mismo, es necesario tener conocimiento de la experiencia de diferentes investigadores o instituciones para la utilización de la nube en modelos numéricos del océano.
A pesar de la relevancia del tema, pocos estudios se han enfocado en el desempeño de modelos numéricos del océano en la nube, la mayoría realizando de evaluaciones para modelos atmosféricos.
Un estudio para un modelo numérico del océano es realizado por Riha (2017), el cual utiliza ROMS en los servicios de AWS, evaluando el desempeño de dos instancias diferentes para una simulación paralelizada con diferentes distribuciones, calculando el precio monetarios del periodo de adaptación de las diferentes simulaciones. Concluyendo la carencia de sentido que tiene el estudio si es que no se llega a comparar con un equipo propio.
En otro estudio de mayor complejidad, Jung et al. (2021) evaluaron el rendimiento de contenedores con ROMS 3.6 en clúster en la nube pública y privada. Basado el la reproductibilidad y portabilidad que brinda Kubernetes, esto se ejecuta en un clúster de AWS utilizando con MPI para paralelizar, con MV conectados mediante *InfiniBand*. Los resultados obtenidos sugieren una buena reproducibilidad de un modelo numérico del océano. Este enfoque abre una posibilidad de implementar este servicio de manera más sencilla incluso para un uso de HPC.
La utilización de contenedores de ROMS también se aplica en Cao et al. (2021), donde se realiza una comparativa para simulaciones entre clúster Kubernetes, clúster HPC y VM HPC. Utilizando servicios de nube para el clúster Kubernetes alojados en Quan-Cloud de *Shandong Computer Science Center2* ubicada en

China.
Un ejemplo más actual de la utilización de la nube es *Coastal Modeling Cloud Sandbox*(Integrated Ocean Observing System, 2023), un entorno que proporciona un marco para desarrollar, modificar y ejecutar modelos enfocados en el pronóstico en entornos de la nube utilizando AWS, *Terraform* y *CloudFlow*, para el desarrollo de cuatro modelos ROMS, SCHISM, ADCIRC y FVCOM. Este entorno es desarrollado por *Integrated Ocean Observing System*(IOOS) perteneciente a la Oficina Nacional de Administración Oceánica y Atmosférica es una agencia científica del Departamento de Comercio de los Estados Unidos (NOAA), enfocada en el desarrollo de nuevas herramientas y pronósticos para la generación de información sobre los océanos, las costas y los grandes lagos, esto haciendo uso de la nube y herramientas que permiten solicitar infraestructura a través de códigos. Otros problemas que han sido afrontados con la herramienta que ofrecen la nube, es el manejo de datos a gran escala y la creación de repositorios. Realiza una una implementación experimental de JupyterHub, Dask y XArray en Google Container Engine (GKE) para apoyar el análisis de datos atmosféricos y oceanográficos en grandes conjuntos de datos(Abernathey et al., 2021). Este proyecto es financiado por *NSF Earth Cube*, perteneciente al gobierno de Estados Unidos.
Existen también proyectos como LiveOcean(Fatland et al., 2016), este es destinado a ayudar a la mitigación de la acidificaron de los océanos en la industria de mariscos en el noreste del Pacífico de los Estados Unidos, donde utilizan el resultado de un modelo acoplado(Roms, WRF y SWAN) que se ejecuta un un clúster propio para generar una salida de pronóstico de 72 horas en una página WEB alojada en la nube Microsoft Azure. Esto para afrontar el principal desafío que es el la cantidad de datos generados y la plataforma de acceso abierta a un público general.

Por parte del área de modelos numéricos atmosféricos, tenemos diferentes estudios también relacionados al rendimiento del cómputo de la nube, pero enfocado en su mayoría al área de pronóstico.
Molthan et al. (2015) realizan pruebas y muestran la factibilidad del uso de WRF en entornos de la Nube de Amazon(AWS) con la utilización de Amazon Elastic Compute Cloud(EC2) generando un clusters para pronósticos con un valor del rango de 40 a 75 USD para un pronóstico de 48 horas, donde el artículo fue principalmente financiado por la NASA.
Siuta et al. (2016) compara un hardware propio con los recursos proporcionados

por los servicios de cómputo de Google Cloud Plataform(GCP) y obtienen una respuesta de viabilidad económica, este estudio financiado principalmente por BC Hydro, Mitacs y *Natural Science and Engineering Research Council of Canada* (NSERC).

Chen et al. (2017) utiliza los servicios de AWS EC2 para ejecutar el modelo climático global CESM (Gent et al., 2011) para evaluar el rendimiento comparado con un servicio de clúster HPC local, teniendo resultados rentables y escalables para la comparación, estudio financiado por *U.S. Department of Energy.*

Zhuang et al. (2019) utiliza los servicios de AWS donde se ejecuta el modelo 3D global GEOS-Chem de química atmosférica almacenado en un contenedor Singularity para clúster, obteniendo una buena accesibilidad para el modelo, pero encontrando problemas de latencia para aplicaciones MPI y que los costes son acorde al usuario. Este proyecto financiado por la *NASA Earth Science Division.*

Chui et al. (2019) explora formas de reducir los costos de la nube para la predicción numérica del tiempo en tiempo real con WRF, y una de las formas es la compresión de archivos resultantes por cobros para la salida de los datos, esta evaluación en GCP. Estudio también financiado por BC Hydro, Mitacs y NSERC

También dentro de la comunidad de Microsoft Azure existe información para correr modelos como WRF, artículos como el de Garvey (2020)(empleado de Microsoft en Microsoft Tech Community), realiza pruebas de WRF v4 en MV HPC, específicamente HBv2, lineas de HPC de Azure. Corriendo una simulación del huracán María para resolución de 1 km, utilizando OpenMP y MPI.

En base a lo expuesto es necesario analizar y generar más contenido que permita evaluar, desde una perspectiva monetaria y de rendimiento de manera general, el uso de la nube en la implementación de modelos numéricos del océano, incluyendo la implementación diferentes herramientas a utilizar. También considerando que actualmente se han realizado mayoritariamente estudios sobre la evaluación de servicios de nube para la ejecución de modelos numéricos en el área de pronósticos para la atmósfera, es una buena oportunidad para evaluar la aplicación de esta tecnología en el ámbito de los modelos numéricos del océano.

Capítulo 2

Objetivos

Objetivo General

Estudiar la implementación de un modelo numérico del océano en un servicio de cómputo en la nube.

Objetivos Específicos

1. Realizar simulación de características reales en el servicio de cómputo en la nube.
2. Detallar el procedimiento para implementar, evaluar desempeño, costes de la aplicación del modelo numérico en la nube.
3. Comparar el procedimiento con otras alterativas de cómputo de alto rendimiento.

Capítulo 3

Metodología

3.1. Selección

Primeramente, se utilizará un proveedor recursos de la nube Microsft Azure (a pesar que se realiza una cotización previa, Anexo A7.1), se utilizará el servicio de Máquina Virtuales, donde se seleccionará la MV más adecuada para realizar la instalación, para luego detallar desde el proceso de creación de la MV, la instalación de CROCO y finalmente la ejecución de la simulación.

3.2. Azure

A pesar de que de los costes asociados, ventajas y desventajas al momento de seleccionar un proveedor, lo más importante es tener un respaldo de experiencia propia y la oportunidad de tener un asesoramiento al momento de utilizar este tipo de servicio, ya que al ser usuarios nuevos no se tiene conocimiento de cobros asociados y tipos de servicios que se ajusten a nuestro trabajo en la nube. Teniendo en consideración lo mencionado previamente, se realizarán pruebas con el proveedor de recursos de la nube Microsoft Azure, debido a la experiencia en trabajos previos con las plataformas de Microsoft.

Para la creación de una MV en Azure solo tenemos que tener una cuenta creada en Azure. Para crear una cuenta debemos ingresar a la página principal de Azure (https://azure.microsoft.com/es-es/free/free-account-faq/) y seguir los pasos indicado en el anexo A2. También se hará uso del periodo de prueba gratuito de Azure de 30 días, en el cual nos proporciona 200 USD para poder realizar pruebas,

con un límite de 4 vCPU para utilizar en MV.

3.2.1. Elección MV

Dentro de lo variado que puede ser las opciones para nosotros adquirir una máquina virtual, tenemos opciones muy básicas hasta opciones de cómputo de alto rendimiento (HPC). Para este trabajo estaríamos centrados en el cálculo de las variables de salida del modelo, por lo que tendremos que seleccionar una MV dedicada a optimizar los procesos. Azure brinda una clasificación de las MVs(se pueden ver clasificaciones en el Anexo A1) donde se puede seleccionar según el uso destinado. Como se puede ver en la figura 3.2.1.

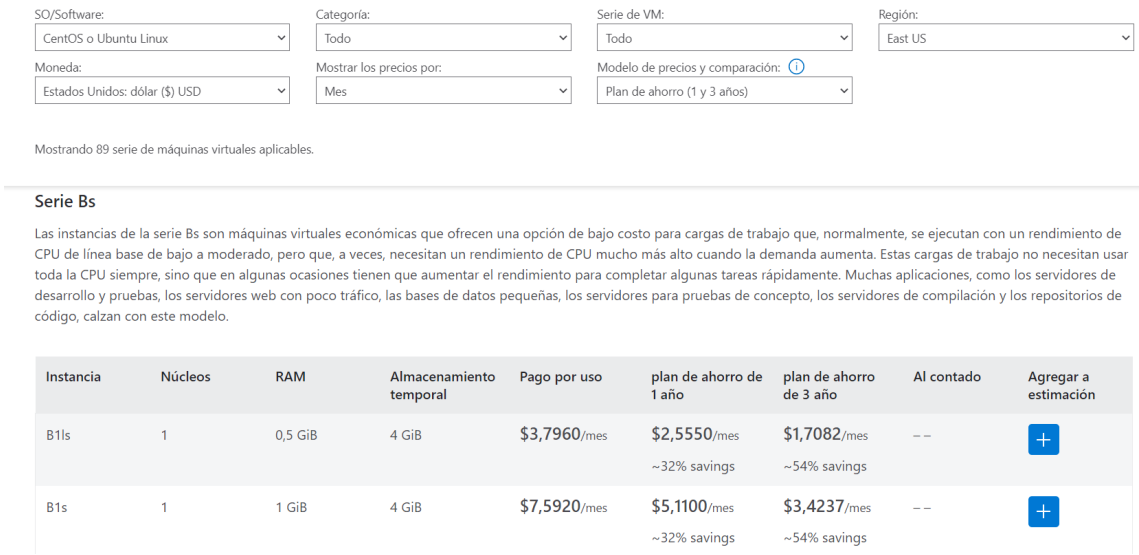


Figura 3.2.1: Página de Microsoft Azure donde se estima el valor de la clasificación de las instancias

Se utilizará las opciones de proceso/cómputo optimizado que ofrece Azure son las series F(Fsv2 y FX). Se selecciona para las series Fsv2 destinadas para proceso y cargas de trabajo de procesamiento vectorial en operaciones de punto flotante simple y doble. Donde cuenta con procesadores Intel® Xeon® Platinum 8272CL (Cascade Lake) de segunda generación y procesadores Intel® Xeon® Platinum 8168 (Skylake). Cuenta con una velocidad de reloj Turbo sostenida de 3,4GHz y una frecuencia turbo máxima de un solo núcleo de 4,3GHz. Para características generales ver Tabla 3.2.1.

Tamaño	vCPU	RAM: GiB	Almacenamient (SSD) GiB	Máx. de Discos
Standard F2s_v24	2	4	16	4
Standard F4s_v2	4	8	32	8
Standard F8s_v2	8	16	64	16
Standard F16s_v2	16	32	128	32
Standard F32s_v2	32	64	256	32
Standard F48s_v2	48	96	384	32
Standard F64s_v2	64	128	512	32
Standard F72s_v22, 3	72	144	576	32

Cuadro 3.2.1: Máquinas virtuales de la serie Fsv2 y sus características generales.

De esta se seleccionará la de tamaño **Standard F4s_v2**, ya que es el límite máximo de vCPU permitido para la prueba gratuita de Microsoft Azure.

3.3. Eficiencia y *Speedup*

Para poder analizar el desempeño de CROCO en la nube, se realizará un escalamiento de núcleos, esto para estimar la eficiencia y *Speedup*, que nos ayudarán a evaluar el desempeño de nuestra simulación(4.2.2).
El escalamiento hace referencia a como se comporta un programa paralelo(que utiliza más de un procesador para ejecutar un trabajo) al aumentar el número de procesadores. Donde se dice que un sistema es escalable para un determinado rango de procesadores $[1...n]$, si la eficiencia E(n) del sistema se mantiene constante y en todo momento por encima de un factor 0.5, donde la eficiencia varía entre 1 y $\frac{1}{n}$. Normalmente todos los sistemas tienen un determinado número de procesadores a partir del cual la eficiencia empieza a disminuir de forma más o menos brusca. Un sistema es más escalable que otro si este número de procesadores, a partir del cual la eficiencia disminuye,

es menor que el otro (Flores Gil, 2004). Aunque cabe destacar que el límite por el cual se considere eficiente dependerá netamente del entorno a trabajar y nuestros objetivos, el valor bajo 0.5 puede ser considera no eficiente en otros casos.

El escalamiento o *speedup*($S(n)$), donde n es el número de procesadores) nos indica la ganancia que obtenemos mediante la paralelización. Este es obtenido entre el tiempo de ejecución de nuestro programa de manera secuencial (uso de un solo núcleo) y el tiempo de ejecución al ir aumentado los núcleos(Flores Gil, 2004):

$$Speedup = S(n) = \frac{T_{original}}{T_{mejora}} \tag{3.3.1}$$

Y la eficiencia del sistema($E(n)$) se define como la porción útil del trabajo total realizado por n procesadores y la podemos obtener de la siguiente forma:

$$E(n) = \frac{S(n)}{n} \tag{3.3.2}$$

Esto será realiza realizado con el fin también de abaratar costes con un resultado de simulación en el menor tiempo, esto teniendo en cuenta que a mayor el poder de la MV mayor es el costo, pero podríamos hacer uso de esta en un menor tiempo, frente a una de menor potencia y coste, pero que se usa en un mayor tiempo, considerando que se hace un cobro por uso.
Estimando la eficiencia y *speedup* puede obtener un gráfico similar a la figura 4.2.3.

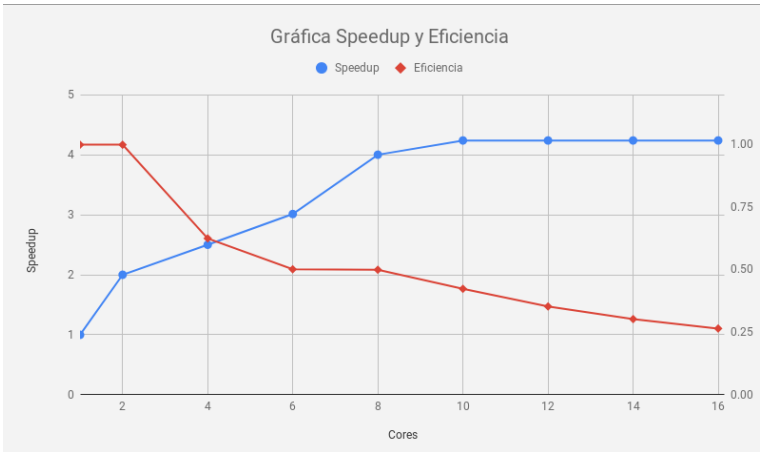


Figura 3.3.1: Ejemplo de gráfica escalamiento con Speedup vs eficiencia para un programa(Nlhpc-Wiki, 2022).

3.4. Paralelización en CROCO

Como se menciona en la introducción, CROCO tiene dos opciones a disposición para generar una paralelización, OpenMP para memoria compartida y MPI para memoria distribuida, sin la posibilidad de utilizar este de forma Híbrida(Como se utiliza en modelos como WRF). Se ofrece una descripción de ambas alternativas, también donde configurar estas opciones:

- **OpenMP:** Es una API que cubre la paralelización dirigida por el usuario, en la que el programador especifica explícitamente las acciones a tomar por el compilador y el sistema en tiempo de ejecución para ejecutar el programa en paralelo(Dagum and Menon, 1998).Este se configura en el archivo cppdefs.h y su división de cálculo se realiza de la siguiente forma:

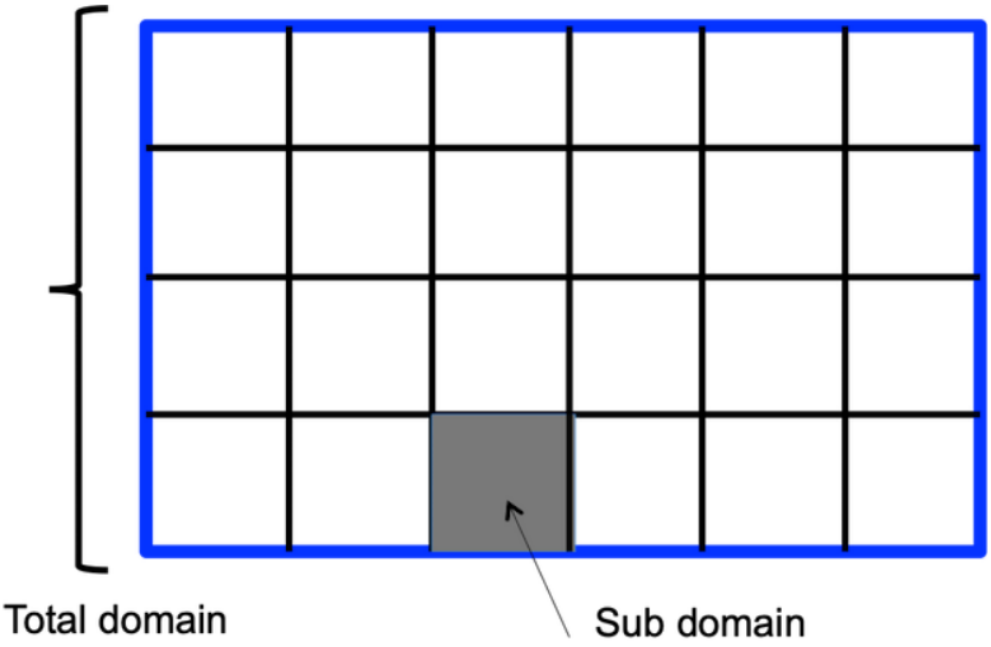


Figura 3.4.1: Descripción gráfica de repartición de cálculo para dominio de un modelo utilizando OpenMP(CROCO, 2020).

luego tendremos que modificar el **param.h**

```
#elif defined OPENMP
parameter (NPP=28)
# ifdef AUTOTILING
common/distrib/NSUB_X, NSUB_E
# else
parameter (NSUB_X=2, NSUB_E=14)
# endif
```

Listing 3.1: Sección de param.h para condiciones si se define OpenMP.

donde **NPP** es número de núcleos,**NSUB_X** es el número de divisiones en la dirección XI, **NSUB_E** es el número de hilos en dirección ETA y **NSUB_X x NSUB_E** tiene que ser un múltiplo de NPP (la mayoría de las veces, establecemos **NPP** = **NSUB_X x NSUB_E**). Este tipo de paralelización esta dedicada a un ordenador que presenta diferentes núcleos de cálculo conectado a un solo banco de memoria RAM(memoria compartida)(CROCO, 2020).

- **Message passing paradigm(MPI):** es una interfaz para máquinas paralelas con memoria distribuida y se ha utilizado ampliamente en sistemas de computación paralela y distribuida(Clarke et al., 1994). Este puede ser definido en el archivo cppdefs.h. Donde sigue una distribución similar de descomposición por grillas del dominio(figura 3.4.1). Esto lo podemos ajustar en el archivo param.h de la siguiente forma:

```
#ifdef MPI
integer NP_XI, NP_ETA, NNODES
parameter (NP_XI=7, NP_ETA=4, NNODES=NP_XI*NP_ETA)
parameter (NPP=1)
parameter (NSUB_X=1, NSUB_E=1)
```

Listing 3.2: Sección de param.h para condiciones si se define MPI.

Donde **NNODES** es número de núcleos,**NP_XI** es el número de baldosas en la dirección XI, **NP_ETA** es el número de hilos en dirección ETA y **NNODES=NP_XI x NP_ETA, excepto cuando se utiliza MPI_NOLAND**, NNP=1 y **NSUB_X - NSUB_ETA**, número de sub-dominios(casi siempre =1)(CROCO, 2020).

Una vez seleccionada una de las dos opciones, solo nos queda compilar el modelo, como para cualquier cambio que se realice en *cppdes.f* o *param.h*.
Para este caso, ya que se utilizará un MV para la realización de la prueba(que es un equipo de memoria compartida) se utilizará OpenMP. Con este tipo de paralelización se realizará las distribuciones de nuestro dominio posibles con 4 vCPU son: 1×1, 1×2, 2×1, 1×4, 4×1 y 2×2.

3.5. Dominio y detalles de simulación

Para visualizar un desempeño más se realizará una simulación realista para un caso en específico de un mes en CROCO, para el dominio siguiente:

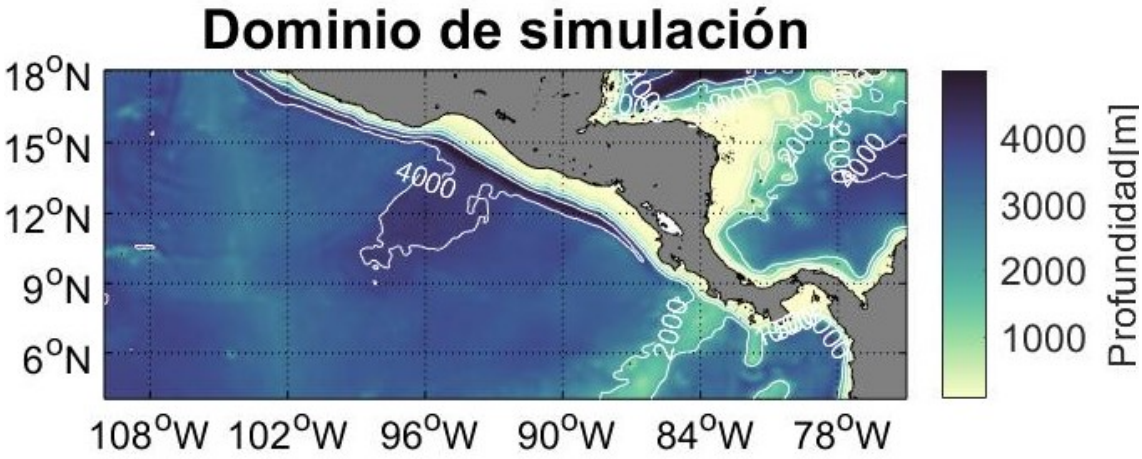


Figura 3.5.1: Dominio a utilizar para la simulación realista en la nube.

De las siguientes características:

Tamaño de las grillas	419x149x42
Píxel de grilla	1/13°∼9[km]
Paso de tiempo	900[s]
Tiempo total	1 día (96 pasos)
Peso de archivo diario con salidas horarias	1.3 GB c/u

Cuadro 3.5.1: Características generales de simulación realista a implementar en la nube.

Se utilizará la versión de CROCO 1.2.1, no se hará uso de CROCO_TOOLS en la MV de Azure.los Archivos de entrada serían generados en la MV LiveCROCO con las herramientas de CROCO_TOOLS en máquina propia, para luego ser subida a la MV del servicio de nube.

Capítulo 4

Resultados

4.1. Creación de una Máquina Virtual

4.1.1. Azure

Para la creación de una máquina virtual en Azure tenemos que tener una cuenta creada en azure.Para crear una cuenta solo queda ingresar a la página principal de Azure (https://azure.microsoft.com/es-es/free/free-account-faq/) y seguir los pasos en el caso de no tener una (Anexo A2).
Una vez creada la cuenta, se generará una suscripción asociada. Esta suscripción es parte de los recursos de Azure, los cuales tiene cuatro niveles de organización, esto se puede ver en al Figura 4.1.1. Este tipo de jerarquía se utiliza para generar flujos de trabajos más claros, sobretodo para empresas o instituciones que requieran una infraestructura en la nube mucho mayor, pero sirve para administrar nuestros recursos y pagos. Ahora bien, dentro del grupo de administración, tenemos lo que denominamos suscripciones, su principal función es generar los cargos del uso de los servicios de Microsoft Azure, la cual están asociados a una tarjeta de crédito o débito.

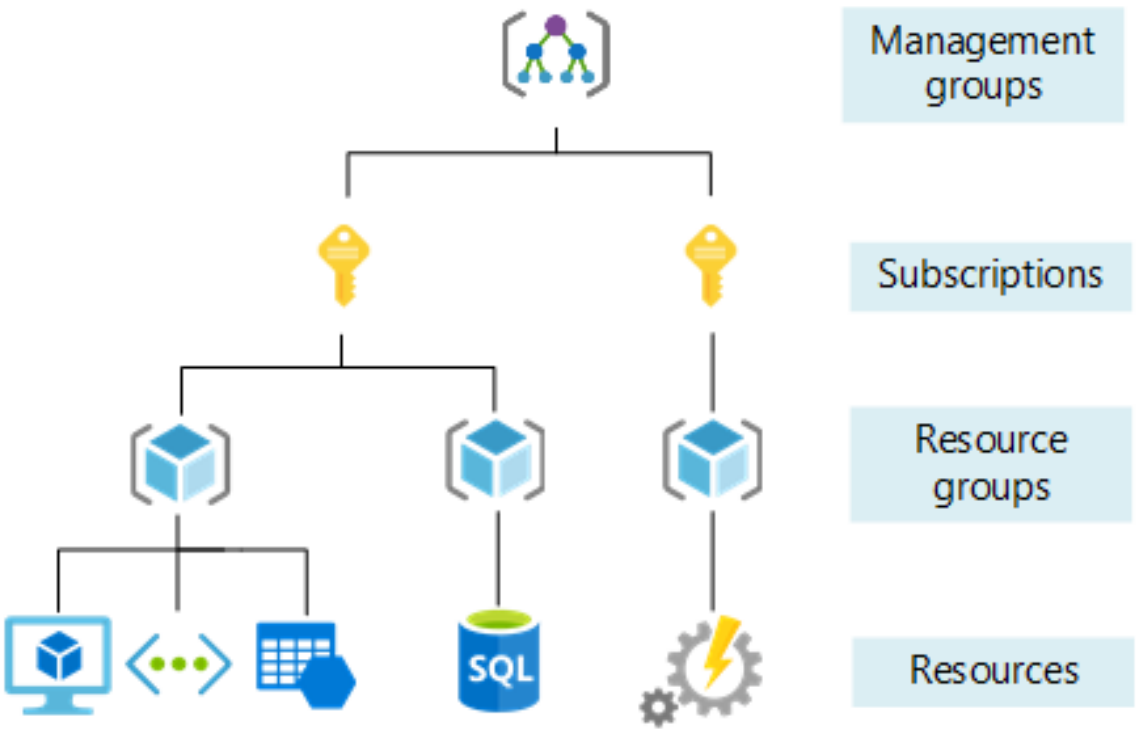


Figura 4.1.1: Jerarquización y organización de recursos en Azure.

El cobro puede variar para las MVs ya que Azure ofrece 3 tipos de formas de pago:

- **Pago Por uso**: el cobro se realiza por solo lo utilizado.
- **Instancias reservadas(solo 1 y 3 años)**: se reserva la MV/Mvs por un periodo de 1 o 3 años, asociado a un descuento respecto al precio de pago por uso.
- **Spot**: Instancias más económicas ya que estan sujetas a ser detenidas.

En el caso de la suscripción gratuita solo se descontará del saldo de 200 USD, una vez pasado los periodos no se seguirá cobrando, solo se tiene que actualizar nuestros datos si es que queremos seguir utilizando el servicio.

4.1.2. Creación de MV(guía de inicio rápido)

Para la creación de una MV, solo necesitamos ingresar al portal de Azure. Primeramente tenemos que acceder al servicio de Azure **Máquinas Virtuales**. Una vez seleccionada este recurso se abrirá una página del recurso donde se puede visualizar las MV creadas y nos permite crear una. Luego seleccionamos **Crear** y luego **Máquina virtual de Azure**.

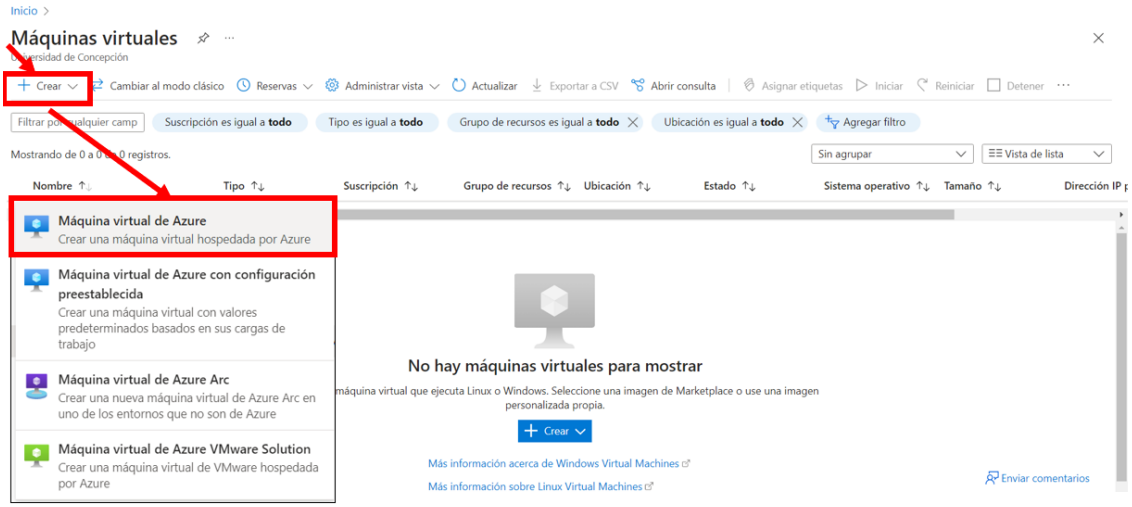


Figura 4.1.2: Ventana de accesos a los recursos de Máquinas Virtuales en Portal Azure.

Se desplegará una página donde podemos configurar los aspectos básicos de nuestra MV. Estos consideran en primera parte, los **detalles del proyecto**, donde se solicita la suscripción y el grupo de recursos (en caso de no tener creado se puede realizar en la misma sección):

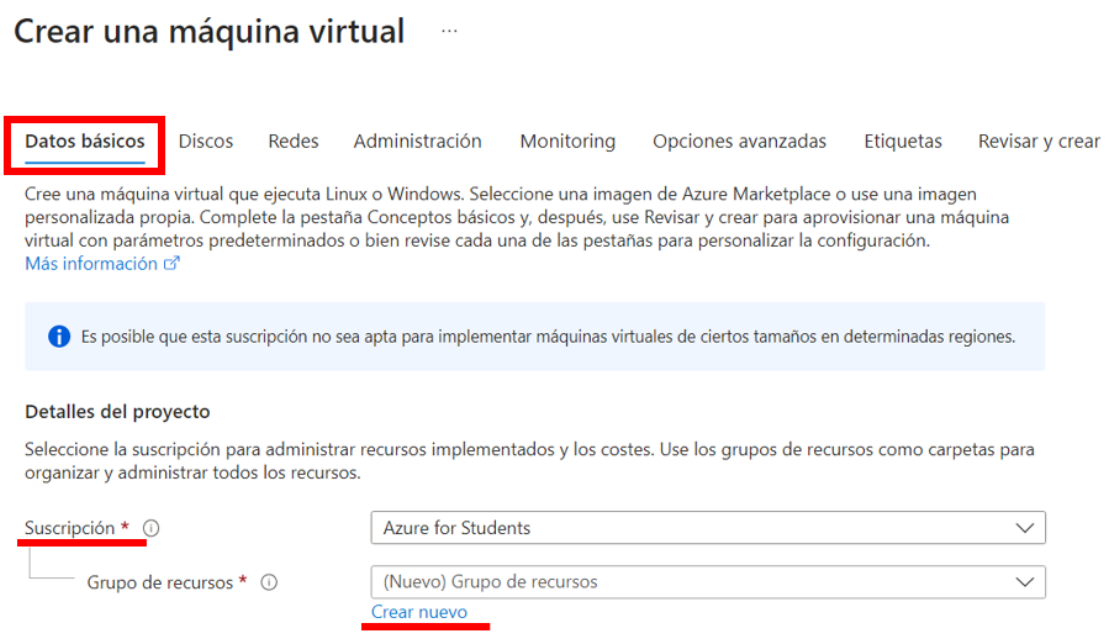


Figura 4.1.3: Sección de asignación de suscripción y grupo de recursos de la MV en Azure.

Ahora también se solicita los **detalles de la instancia** donde tenemos; **Nombre de la máquina**; **Región**, esta puede influir en la latencia de conexión, disponibilidad de series de MV y costes de salida de datos; **Opciones de disponibilidad**, relacionada a redundancia de infraestructura; **Tipo de seguridad**, relacionada a ciberseguridad; Imagen, esta es el sistema operativo seleccionado; **Arquitectura**; **Ejecución de Azure Spot con descuento**, donde estaremos sujetos a la pérdida de infraestructura; **Tamaño**, selección de tipo de serie de MV, numero de vCPU y cantidad de memoria RAM(Mas detalles de series en el Anexo A1).

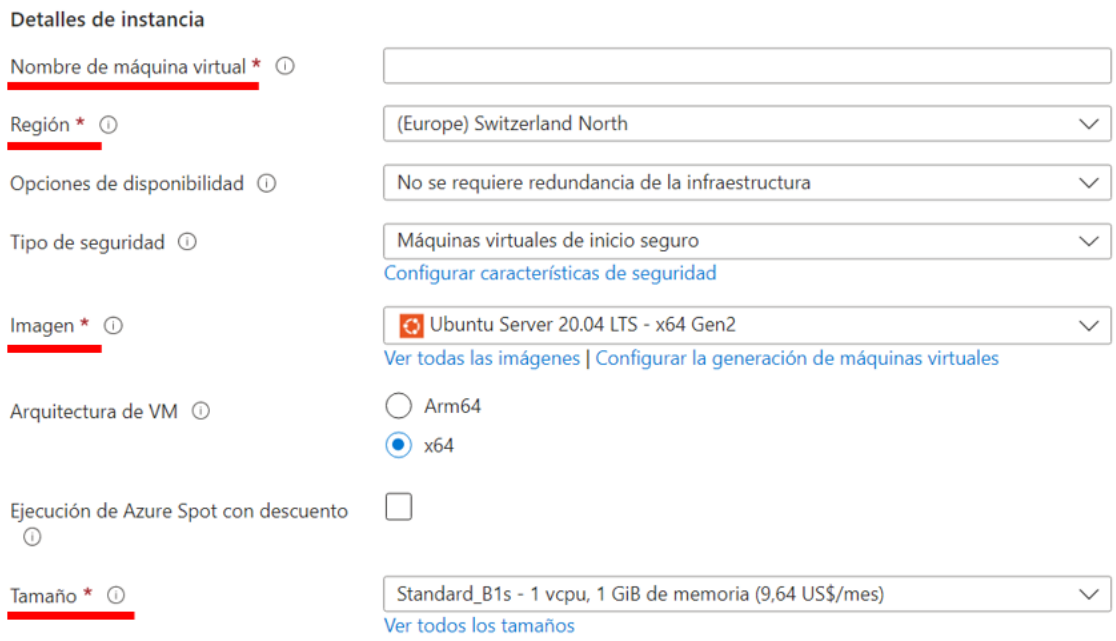


Figura 4.1.4: Designación de detalles de la instancia para la creación de MV Azure.

Luego en la sección **Cuenta de administrador** configuramos el tipo de autenticación, puede ser mediante clave pública SSH o Contraseña, dependiendo de lo seleccionado también tenemos que crear el nombre de Usuario.

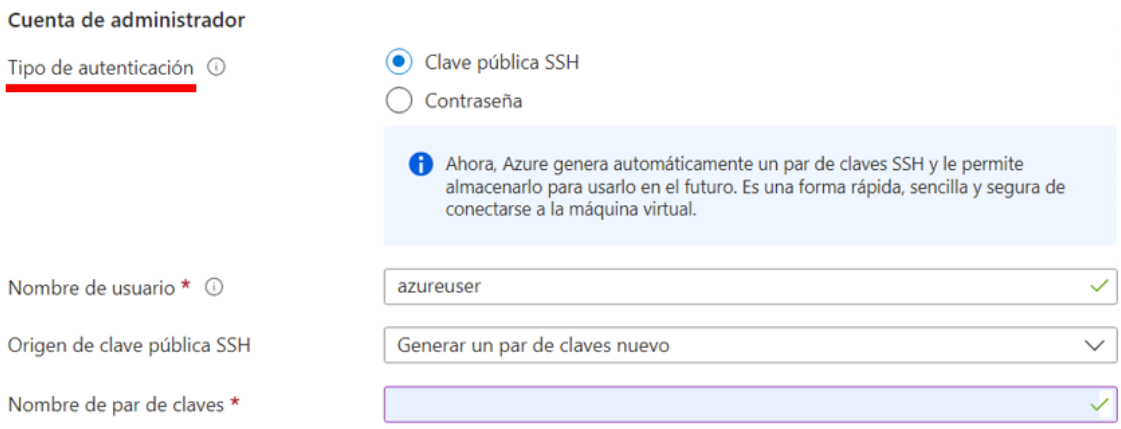


Figura 4.1.5: Sección de opciones de autenticación de una MV en Azure.

Finalmente, solo queda configurar las **Reglas de puerto de entrada**, que son permitir puertos de entrada a la MV que en este caso se selecciona el 22 para conexión con protocolo SSH.

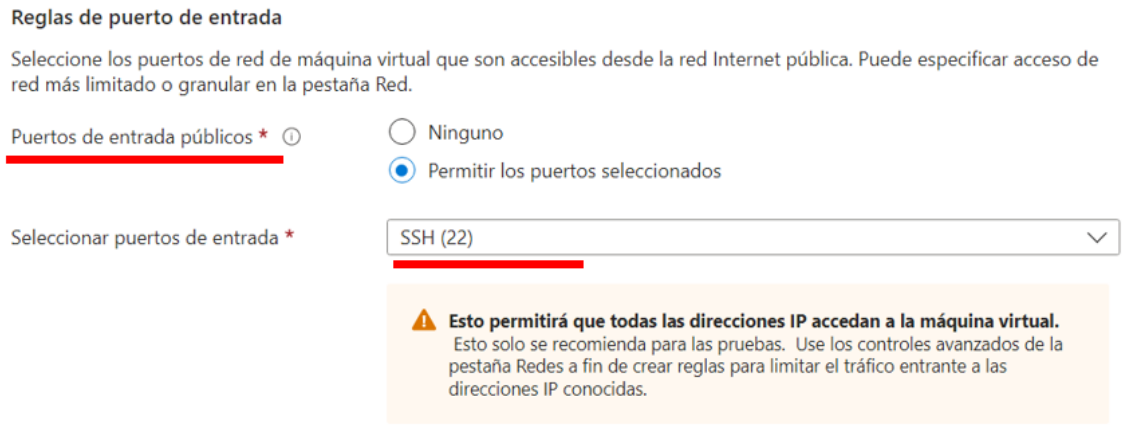


Figura 4.1.6: Sección de ajustes de liberación de puertos de redes.

Otras opciones necesarias también son añadir **Discos**,**Redes**, **Administración**, **Monitoreo**, **Opciones Avanzadas** y **Etiquetas**. Para la creación rápida de una MV solo basta configurar los **Datos básicos**, pero también podemos agregar un Disco de almacenamiento en caso de ser necesario, esto en la opción de Discos. Con todo configurado vamos a, **Revisar y Crear**, donde Obtendremos un resumen con todas las configuraciones seleccionadas y un aproximado del valor que tendrá el uso de nuestra MV.

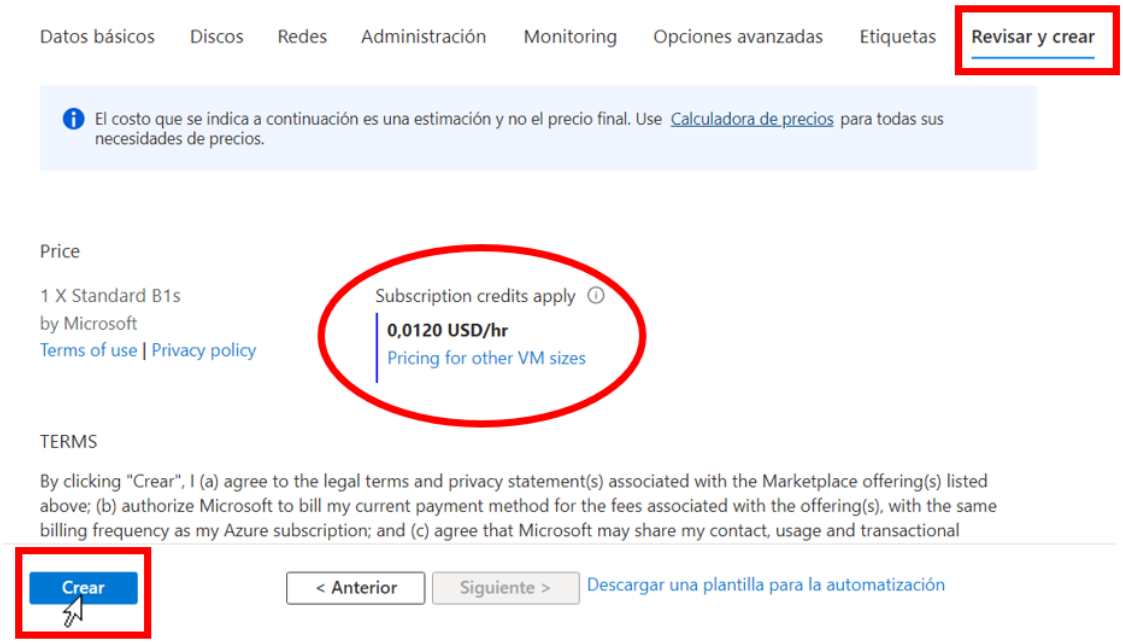


Figura 4.1.7: Resumen de costos para MV de Azure, paso anterior a la creación.

Ahora Azure crea los recursos necesarios como el disco de almacenamiento, una interfaz de red,la dirección IP y finalmente nuestra MV. Donde tenemos que esperar a la creación de estos y dirigirnos al nuestro recurso luego de ser creado.

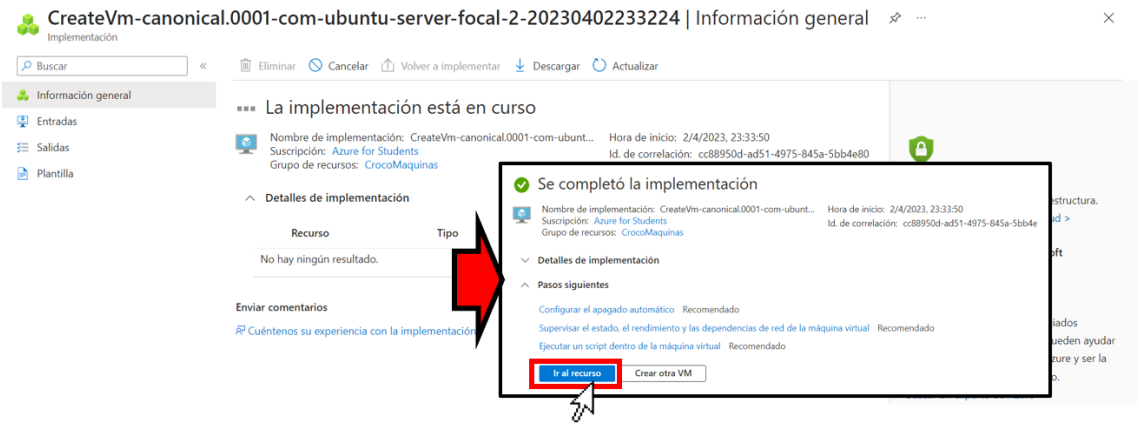


Figura 4.1.8: Sección de de implementación de recurso de MV.

Una vez ya creada nuestra MV, Azure nos proporciona una ventana con un resumen de esta, con todas las configuraciones solicitadas y también la dirección IP de la MV. Dentro del despliegue de información general tenemos diferentes opciones de las cuales podemos destacar: el cambio el tamaño de nuestra MV(ya sea aumentando o disminuyendo), agregar discos de almacenamiento y opciones de conexión a través de protocolos de conexión o también a través de la consola de Azure.

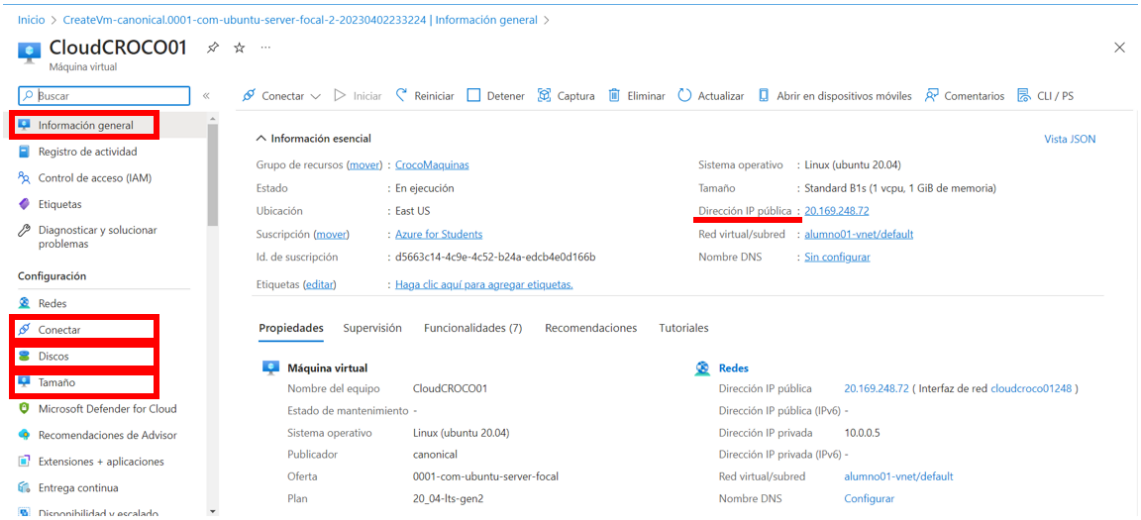


Figura 4.1.9: Sección de información general de una MV en portal de Azure.

4.1.3. Instalación de CROCO

Para poder instalar CROCO, como se menciona en la sección 5, se necesita librerías de Fortran, Netcdf, MPI y compilador de C. Todo esto se puede instalar directo de la terminal, o también podemos utilizar el siguiente código que no facilitará la instalación de todas las librerías y también la descarga de CROCO desde la página web. No será necesario descargar CROCO_TOOLS ya que no necesitaremos generar archivos de entrada, ya que estos se generaron en una máquina propia. El tiempo de instalación para la MV F4s_v2 es de 104 segundos.

```
#!/bin/bash
apt-get update -y
apt-get upgrade -y
apt-get install gfortran -y
apt-get install make -y
apt-get install libopenmpi-dev -y
apt-get install libhdf5-openmpi-dev -y
apt-get install libnetcdf-dev -y
apt-get install libnetcdff-dev -y
apt-get install bc -y
apt-get install git -y
apt-get install curl -y
apt-get install netcdf-bin -y
apt-get install nco -y
apt-get install unzip -y
apt-get install wget -y
apt-get install sudo -y
apt-get clean -y

wget https://data-croco.ifremer.fr/CODE_ARCHIVE
/croco-v1.2.1.tar.gz
gzip -d croco-v1.2.1.tar.gz
tar -xvf croco-v1.2.1.tar
rm croco-v1.2.1.tar
```

Listing 4.1: Instalación de librerías necesarias y descarga de CROCO.

4.1.4. Opciones de conexión

Para poder conectarnos a nuestra máquina tenemos que tener abierto o habilitado un puerto de entrada a la MV. Por defecto y generalmente viene activado el puerto 22 para protocolo *SSH. Secure SHell* es un protocolo que permite crear conexiones seguras entre dos ordenadores, donde la máquina del cliente inicia una conexión con la máquina del servidor mediante una sesión cifrada, imposibilitando que alguien pueda obtener una contraseña o cualquier otro tipo de información que se envíe por la red(Bonet Esteban, 2014). Tenemos las siguientes alternativas:

1. **Claves SSH:**Servicio de Azure donde se designan claves .pem para diferentes grupos de recursos. Se asignar la suscripción y el grupo de recursos. También tenemos que añadir los detalles de la instancia, donde seleccionamos la Región, el nombre del par de claves, y tenemos la opción de que Azure nos genere una clave o se podemos cargar la de nosotros. Esta opción también se puede realizan un vez nosotros creemos nuestra máquina. Teniendo descargado nuestro .pem solo nos aseguramos que tengamos el permiso de lectura para nuestro usuario. Esto se realiza de la siguiente forma:

   ```
   chmod 400 <keyname>.pem
   ```

   Y tenemos que ejecutar el comando en nuestro terminal para conectarnos:

   ```
   ssh -i <ruta de acceso de clave privada> Usuario@IP_MV
   ```

2. **Contraseña:** Generamos una contraseña para nuestra MV al momento de crearla y nos podemos conectar con el siguiente comando:

   ```
   ssh usuario@IP_MV
   ```

   Se nos preguntará la contraseña y luego nos conectamos.

3. **Generar propia clave:**Podemos utilizar nuestra máquina y en la consola ejecutamos ***ssh-keygen***, que nos generará una clave publica, la que tendremos que asociar esta al momento de crear nuestra MV.

También al abrir el puerto 22 tenemos la opción de subir archivos a través del protocolo **SFTP**. Esto lo podemos realizar una vez tengamos configurada las opciones ssh, abrimos la terminal en la ubicación de la máquina donde se encuentren los archivos que se requieran subir y ejecutamos el siguiente comando:

```
sftp usuario@IP_MV
```

Una vez conectado, solo ejecutamos el comando **put** seguido del nombre del archivo, como la siguiente forma:

```
put Nombredelarchivo
```

4.1.5. Almacenamiento en la Nube: Discos

Azure ofrece un servicio llamado Discos, los discos son volúmenes de almacenamiento de nivel de bloque que administra Azure y que se usan con Azure Virtual Machine. Podemos considerarlo como un disco físico en un servidor, pero no hay que olvidar que esta virtualizado. Opciones de Discos de almacenamiento en Azure:

	SSD Premium	**SSD estándar**	**HDD estándar**
Tipo de disco	SSD	SSD	HDD
Escenario	Cargas de trabajo delicadas de producción y rendimiento	Servidores web, aplicaciones empresariales poco utilizadas y desarrollo y pruebas	Copia de seguridad, no crítico, acceso poco frecuente
Tamaño máximo del disco	32 767 GiB	32 767 GiB	32 767 GiB
Rendimiento máx.	900 MB/s	750 MB/s	500 MB/s
IOPS máx.	20.000	6.000	2 000-3000

Cuadro 4.1.1: Disco de almacenamiento disponible para utilizar en una MV con SO para Microsoft Azure(Microsoft-Learn, 2023a).

El rendimiento de los discos se mide en IOPS y estas son las operaciones de entrada y salida por segundo, cada disco tiene diferentes capacidades, como también las MV tienen límites asociados.

4.1.6. Instantáneas

Una de las herramientas útiles al momento ya de tener nuestra MV configurada y realizar réplicas de esta, es generar instantáneas. Esta es una copia completa de solo lectura de un disco duro virtual (VHD). Puede usar una instantánea como copia de seguridad de un momento determinado o para ayudar a solucionar problemas de la MV. Puede tomar una instantánea del sistema operativo o de los discos duros virtuales(Learn-Microsoft, 2023).
Para realizar una instantánea debemos contar con una MV ya funcionando, y también un disco donde almacenar esta instantánea. Para esto Azure cuenta con un servicio denominado **Instantáneas**, donde podemos realizar una instantánea a nuestra MV.

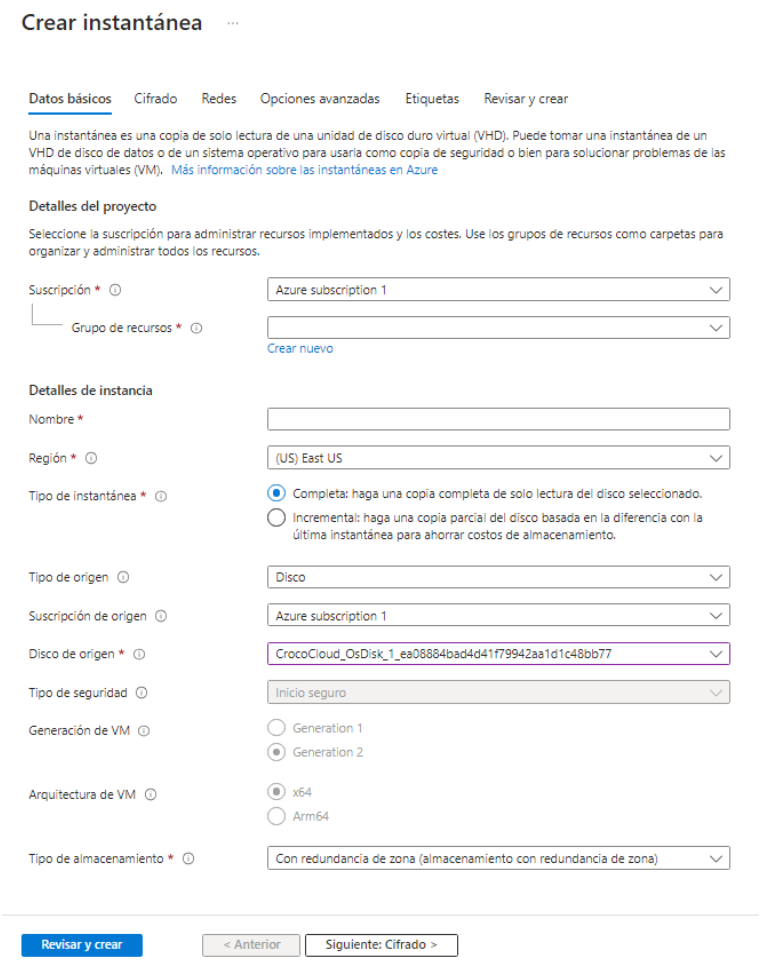


Figura 4.1.10: Ejemplo de detalles básicos al crear una instantánea en Microsoft Azure.

Rellenando los detalles del proyecto (Suscripción y grupo de recursos), detalles de la instancia: Nombre; Región; tipo de instantánea, completa o incremental; tipo de origen, disco o instantánea; suscripción de origen; disco de origen; tipo de almacenamiento, HDD, SSD o con redundancia de zona. Los precios se cobran por GB de base más tiempo de almacenamiento.

4.1.7. Azure CLI

Azure CLI es una herramienta de linea de comandos multiplataforma útil para conectarse a Azure y ejecutar comandos administrativos en recursos desde Azure desde la terminal.
Para la instalación de esto, el equipo de CLI de Azure mantienen un scipt para ejecutar todos los comandos de instalación en un solo paso. Este script se descarga y permite instalar la CLI.

```
curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
```

Una vez iinstalado solo basta ejecutar el siguiente comando para iniciar sesión:

```
az login
```

Se despliega una ventana en el navegador predeterminado de la máquina, donde hay que ingresar con nuestra cuenta de Azure. Una vez listo podremos empezar a ocupar Azure CLI.

4.1.8. Creación de Mv y entorno con Azure CLI

Primero, creamos un grupo de recursos desde la consola de la siguiente forma:

```
az group create --name recursodegrupo --location eastus
```

Designando el nombre y la locación del grupo. Ahora podemos crear una MV con el siguiente comando:

```
az vm create \
    --resource-group recursodegrupo \
    --name nombre \
    --image UbuntuLTS \
    --public-ip-sku Standard \
    --admin-username usuario
```

Donde podemos seleccionar el nombre de nuestro grupo de recurso, nombre de la MV, imagen a utilizar, tipo de ip y nombre de usuario del administrador. Esperamos unos minutos a que se cree el recurso. Obteniendo una salida donde nos mostrará detalles de la MV creada y el estado *running*. También con Azure CLI podemos generar un clúster básico como se ve en el Anexo A7

4.2. Pruebas Micrsoft Azure

4.2.1. MV F4s_v2

La máquina virtual tiene sistema operativo *Linux Ubuntu 20.04*, su ubicación es en *East US*, también para la compilación se utiliza *gfortran*(versión 9.4.0). El coste de este es de 0.16 USD/hr, todo para una configuración básica de una MV que cuenta con 4 vCPU, 8 GiB de RAM y un disco SSD estándar de 30 GiB.

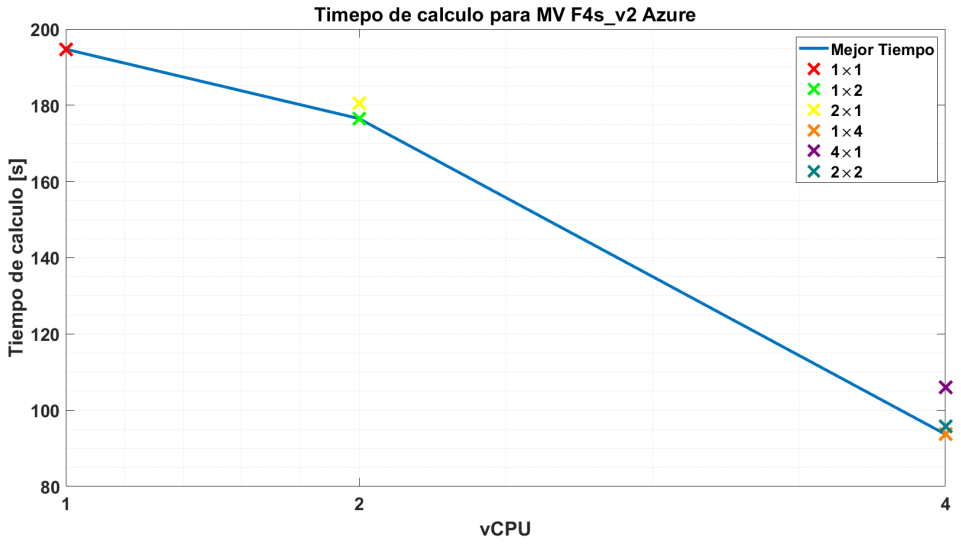


Figura 4.2.1: Tiempo en calculo para simulación de 1 día en MV F4s_v2 de Azure.

En la figura 4.2.1 se tiene un tiempo de 194,7[s] para 1 vCPU. Ahora vemos una disminución a la hora de utilizar 2 núcleos, siendo una diferencia de 18,2[s] y 14,7[s]. Se obtiene un menor tiempo para el uso de 4 vCPU con un tiempo de 93,5[s]. Notamos que el menor tiempo se cumple para las divisiones en la dirección *eta*. Para el único caso de distribución isométrica, no llega a ser menor el tiempo de cálculo por 2[s] por sobre el tiempo menor.

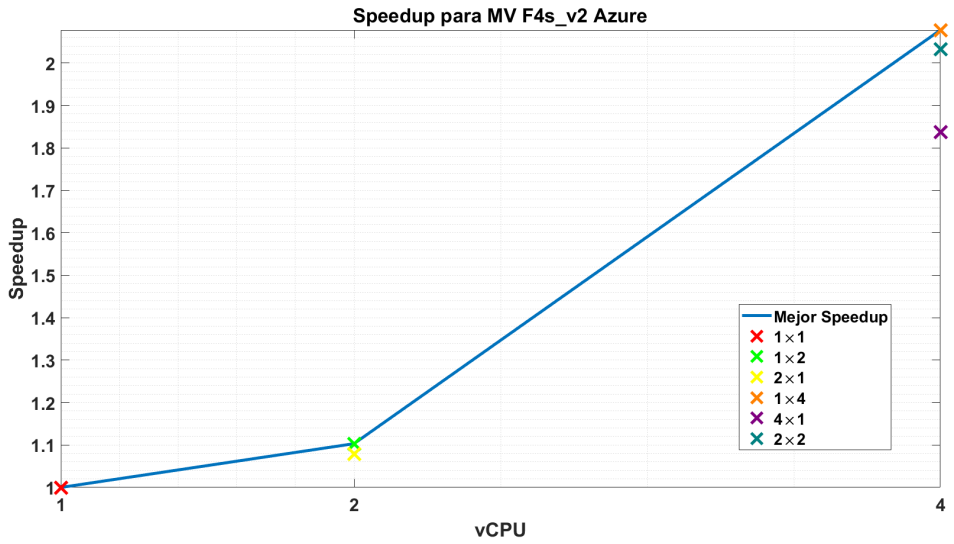


Figura 4.2.2: *Speed-up* para simulación de 1 día en MV F4s_v2 de Azure.

Para la estimación de Speed-up en la figura 4.2.2, se obtiene lo esperado, donde vemos que los resultados de 2 vCPU el tiempo es de 1.09 y 1.07, un valor bastante bajo. También notamos un aumento del *Speed-up*, siendo el mayor para la distribución 1×4. Tenemos el mismo comportamiento que para el gráfico anterior, pero siendo los valores de *Speed-up* más altos para la que tenga mayor distribución en *eta*.

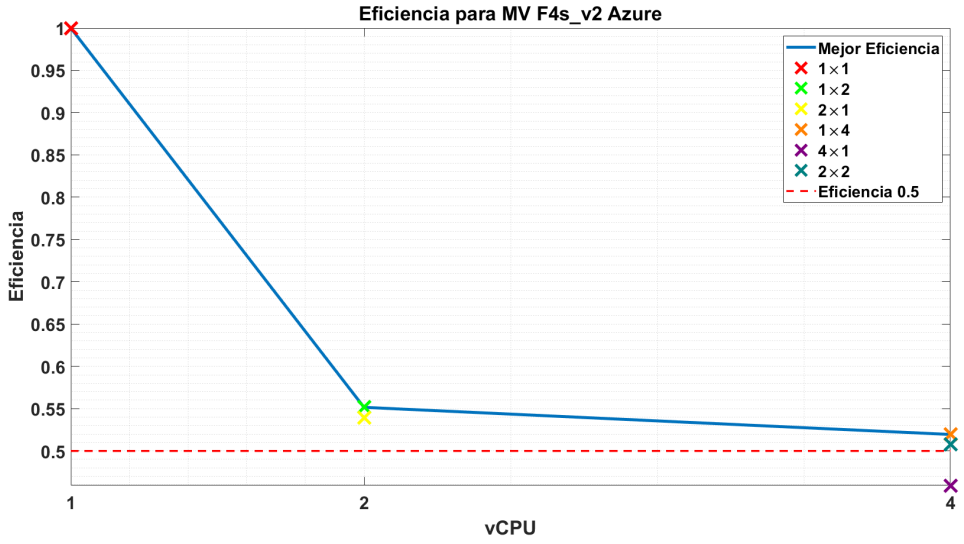


Figura 4.2.3: *Eficiencia* para simulación de 1 día en MV F4s_v2 de Azure.

Para la estimación de eficiencia, tenemos que entre más cercano al valor a 1 y dependiendo del caso, se estima el valor por sobre 0.5 para que el proceso sea considerado eficiente para la cantidad de procesadores utilizados para la paralelización. Ahora en cuanto a los valores obtenidos de eficiencia, y considerando un valor de 0.5 como mínimo de eficiencia, tenemos que solo una prueba deja de ser eficiente para la paralelización, que en este caso es la distribución 4×1. También se ve que los valores más altos de eficiencia son lo que cuentan con más divisiones en dirección *eta*. Para este caso con 4 núcleos distribuidos isométricamente(2×2) o en dirección *eta* siguen siendo eficientes. Se obtienen los resultados de relación inversa entre *Speed-up* y eficiencia, estos se puede observar en la figura 4.2.4.

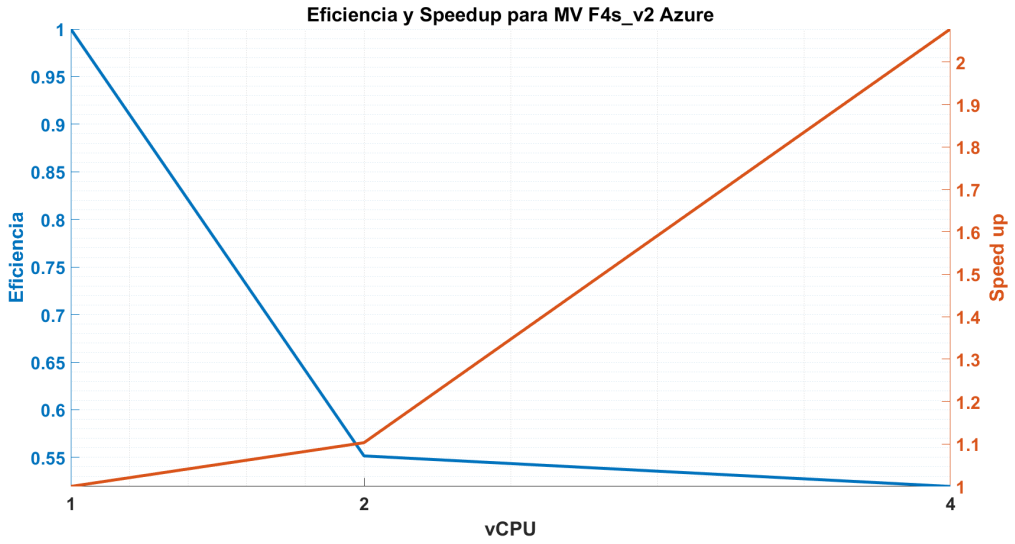


Figura 4.2.4: Relación entre *eficiencia* y *Speed-up* para simulación de 1 día en MV F4s_v2 de Azure.

4.2.2. Costo

Para la realización de las pruebas solo se hizo uso de 0.39 USD del crédito gratuito. Este valor considera la instalación de CROCO(con requisitos de paquetes), la subida de nuestros archivos iniciales y también la realización de otras pruebas (MV D4as_v4 de azure sección A6), aunque no incluye valores de descarga de los archivos de salida. Podemos ver en la figura 4.2.5, que 0.28 USD son costes de almacenamiento, 0.06 USD de costes de redes virtuales y 0.05 por las MV. Esta información se obtiene con el servicio de **Administración de costos + facturación**.

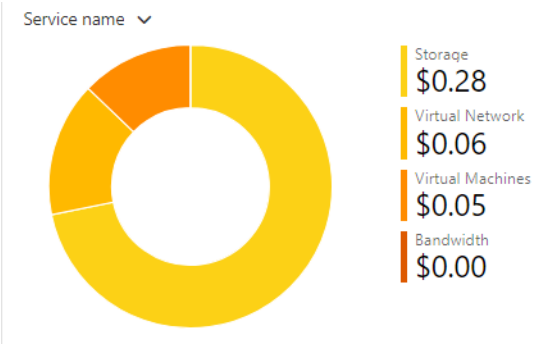


Figura 4.2.5: Análisis de costos asociados a los servicios de Microsoft Azure para las pruebas realizadas.

Capítulo 5

Discusión

5.1. MV Microsoft Azure

La implementación de una MV puede ser una buena solución si es que se requiere de recursos informáticos de manera rápida. El aprovisionamiento de una MV, teniendo ya una cuenta de Azure puede tomar unos minutos, este tiempo se puede simplificar si es que utilizamos Azure CLI, utilizando códigos previamente preparados para una solicitud de MV, automatizando el proceso. También se pueden realizar instantáneas de MV donde nosotros tengamos la configuración de nuestro modelo ya realizada, reduciendo aún más el tiempo de implementación de CROCO.

Se ha realizado con éxito la simulación realista y se observa que hay un rendimiento mejorado al aumentar los núcleos de cálculo para paralelización con una división del dominio en la dirección *eta*. Este resultado es inesperado, ya que generalmente se espera que haya un mayor tiempo de comunicación entre las subgrillas cuando hay intersecciones más grandes, esto debido al mayor tamaño en la dirección latitudinal del dominio. Sin embargo, en este caso para una mayor cantidad de divisiones en dirección *eta* se obtiene un mejor rendimiento, incluso en comparación a la distribución isométrica (2×2).

Al momento de utilizar 4vCPU, la distribución 2×2 y 1×4 están por sobre el umbral de 0.5 pero por una mínima diferencia (0,008 y 0,02 respectivamente), mientras que la distribución 4 deja de ser eficiente, lo que podría dar un indicio que para este tipo de simulación donde no se utiliza ningún tipo de módulo extra y solo se graban las variables de CROCO por defecto de forma horaria, no es

necesario una mayor cantidad de vCPU, aunque esto desde el punto de vista de la eficiencia. Otro podría ser el caso donde deja de ser eficiente la cantidad de núcleos, pero el tiempo de simulación es menor, lo que podría implicar un menor gasto.
Otro punto importante a destacar es que al calcular la eficiencia se ve afectada por el tiempo requerido de grabado de los resultados. Si se desea a medir la eficiencia de forma más precisa con el objetivo obtener el rendimiento real de la MV, se pueden realizar pruebas sin el grabado de resultados. En el anexo A5 se presenta un ejemplo en GCP en el cual se obtienen valores de eficiencia muy elevados para 4vCPU de esta misma simulación. Aunque en la práctica, este tipo de análisis es totalmente teórico, ya que el objetivo principal es obtener resultados de la simulación. Debido a esto se pueden considerar las siguientes opciones para mejorar la eficiencia:

- Aumentar el intervalo de tiempo entre los pasos de guardado: Esto implica una reducción de la sobrecarga de escritura del disco, disminuyendo el tiempo de cálculo.
- Mejorar el rendimiento del disco: Como se menciona en los resultados sobre los discos, estos miden su rendimiento en IOPS, por lo que tendríamos que evaluar el desempeño del disco que se está utilizando.

5.1.1. Costos

En relación al costo de las pruebas tenemos que considerar qué a pesar de ser muy bajo, se tenia un conocimiento ya de como poder instalar el modelo en un máquina, esto mediante el código para la instalación de CROCO. También se realizaron los archivos iniciales en máquina propia, lo que reduce bastante los costes. Por lo que el realizar este tipo de pruebas (considerando la MV de 4vCPU solamente) es bastante económico. Especialmente si se utiliza el crédito de prueba de 200 USD, es una buena alternativa para realizar pruebas y adquirir experiencia.
Otro factor que reduce los costes es que no se realiza descarga de los archivos resultantes, por lo que no se genera un gasto de transferencia de datos. Si es que se hubiese realizado la descarga, esta cuenta con un tamaño de 1,3 GB, no considera un gasto, debido a que el límite de transferencia de datos es de 100 GB/mes.

5.1.2. Estimación

Considerando que el tiempo de cálculo y el grabado de las salidas tiene un crecimiento lineal, podemos realizar una aproximación de manera gruesa para una simulación climatológica. Supongamos que se quisiese calcular 20 años de simulación, solo guardando las variables que vienen por defecto para CROCO con una salida horaria. Solo nos queda estimar un valor de tiempo de cálculo.

Distribución	**Tiempo para 1 día [s]**	**Tiempo 20 años [hrs]**	**Costo simulación(USD)**
1×1	194.7	394.8	66.7
1×2	176.5	357.9	60.8
2×1	180.5	366	61.8
1×4	93.7	190.0	32.1
4×1	106	214.9	36.3
2×2	95.8	194.2	32.8

Cuadro 5.1.1: Costo y tiempo de cálculo para una simulación de 20 años $\mathrm{F4s}_v 24 vCPU$.

Ahora también podemos suponer el rendimiento de para una máquina Fsv2 de 2vCPU que tiene un coste de 0.08 USD/hr

Distribución	**Tiempo 1 día [s]**	**Tiempo 20 años [hrs]**	**Costo simulación (USD)**
1×1	194.7	394.8	33.1
1×2	176.5	357.9	30
2×1	180.5	366	30.7

Cuadro 5.1.2: Costo y tiempo de cálculo para una simulación de 20 años Fsv2 2vCPU

Si bien se tiene un precio similar a los resultados utilizando 4vCPU, la diferencia entre el tiempo de cálculo entre 2 a 4 vCPU para la distribución de menor tiempo es de 167 horas y costo de 2.1 USD, costo bastante bajo para la gran cantidad de tiempo ahorrado.

También podemos realizar una comparación de una prueba con otro tipo de MV, D4s_v2 que tiene un costo de 0.875 USD/hora(Detalles de la prueba en Anexo A6). Realizando el mismo ejercicio:

Distribución	Tiempo día [s]	Tiempo 20 años [hrs]	Costo simulación (USD)
1×1	115.2	233.6	51.3
1×2	104.7	212.3	46.7
2×1	115	233.2	51.3
1×4	57	115.5	25.4
4×1	68.8	139.5	30.6
2×2	63.6	128.9	28.3

Cuadro 5.1.3: Costo y tiempo de cálculo para una simulación de 20 años D4as_v4 4vCPU

Si comparamos la distribución con el desempeño de las MVs, notamos que para el caso de 1×4 de la serie D4as_v4 logra un mejor desempeño con una diferencia de tiempo 75.5 hrs. y de coste de 6.7 USD con respecto a la serie Fsv2 . Ahora si bien analizamos la diferencia de valores de la eficiencia es de 0.01, ambos por encima del umbral 0.5, por lo que en este caso sería más conveniente utilizar una serie de MV más costosa pero para un menor tiempo y valor total.
Ahora también tenemos que considerar que esta es una aproximación de forma muy gruesa. Podría no ser preciso a la hora de extrapolar a otro tipo de simulaciones, esta presenta una resolución alto, no se realiza la aplicación de ningún otro módulo de CROCO o acople de otro modelo. Si consideramos estos factores, esto puede aumentar el gasto, cómo por ejemplo, algunos módulos aumentan el tiempo de compilación, aunque para esto se podrían generar pruebas probando otro tipo de compilador más eficiente y reducir el tiempo. Por último también considerar los archivos de salida pueden generar otro gasto de salida por transferencia de datos no contemplado si es que terminan siendo archivos muy pesados.

5.2. Alternativas para reducir costes

Una alternativa también que se implementa para generar un ahorro, es arrendar spot Machines tal como lo hace Chui et al. (2019), que utiliza estas MV que son propensas a ser detenidas en AWS(EC2) para la implementación del modelo numérico WRF. Un tipo de MV que utilizo es la *c5n.18xlarge* (36 núcleos) cuyo coste fue 1,17 USD por hora para el año 2020, siendo que el valor para la misma MV por uso es de 3.88 USD por hora, donde a pesar de seleccionar este tipo de MV no tuvieron problemas de detención. Para el caso de Azure , también cuenta la opción de spot, donde el valor por eso horario sería 0.0341 USD para la máquina la cual se realizaron pruebas(Fs4_v2), aproximadamente un 79 % de ahorro respecto al servicio que se utilizó.
Este tipo de MV que acepta interrupciones también fue evaluada por Chui et al. (2019) para modelo numérico WRF en el proveedor de servicios de nube GCP, donde prueba un método para que la simulación se reinicie automáticamente desde su último archivo creado si es que se detiene la MV, presentando algunas complicaciones pero sin comprometer el resultado de simulación. Sumado a esto se estudian diferentes configuraciones del modelo, análisis de rendimiento de compiladores, reducir la frecuencia de guardado de salidas y también la compresión de archivos de salida para la reducción de costes, esto último en relación de postprocesamiento realizados en GCP (en casos donde no se requieran las salidas completas) lo que podría ser una solución en el caso tener salidas con un peso mayor a 100 GB en un mes, por lo menos en Azure.

5.3. NLHPC

En nuestro país contamos con un servicio importante a la hora de solicitar recursos computacionales para el avance de la investigación científica, este es el Laboratorio Nacional de Computación de Alto Rendimiento (también conocido como NLHPC), este brindando infraestructura a la comunidad científica, universidades e instituciones públicas y privadas, este cuenta con dos clústers, cuyas características y funcionamiento se pueden revisar en el Anexo A3 .
Se puede solicitar una cuenta a través de la página del NLHPC, donde nos piden datos generales, propuestas, escalamiento de nuestra aplicación, etc. Aunque principalmente se evalúa la calidad de la propuesta, el currículo del investigador,

experiencia en HPC y aportes monetarios. Este último punto pide ser justificado y que tenga relación con el tiempo de cálculo que se utilizará, por lo que podemos decir que en sí el servicio no es gratuito del todo, ya que influye en el método de evaluación para solicitud de cuenta, lo que no quiere decir que no será aprobada una solicitud sin un aporte monetario.
Un problema que se visualizó al momento de investigar y utilizar los servicios de NLHPC es la gran solicitud que tiene este servicio y que puede generar problemas si es que necesitamos respuesta inmediata por parte del clúster, es más, al momento de escribir este documento, no hay nodos libres en el clúster de Leftraru y 4 nodos en Guacolda.

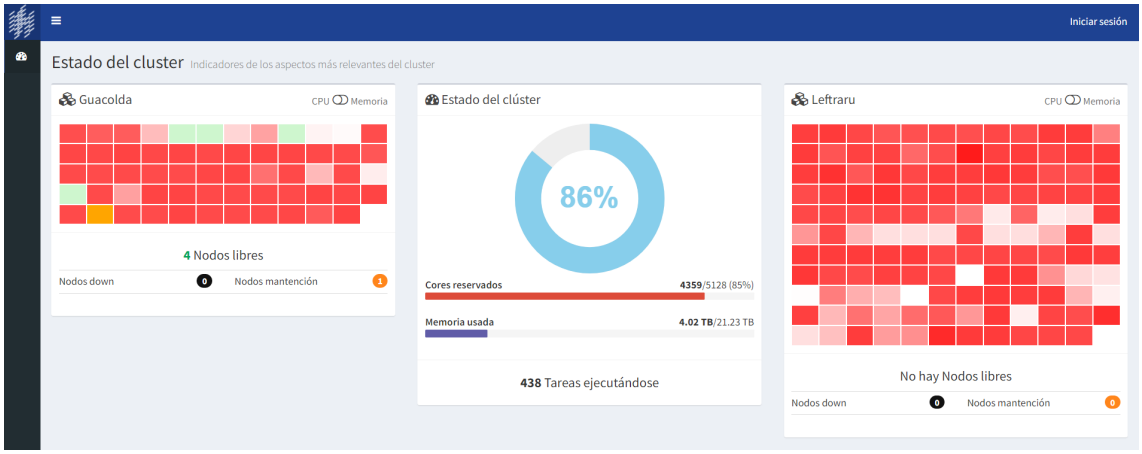


Figura 5.3.1: Estado de la página https://dashboard.nlhpc.cl/ para el día 8 de junio del año 2023.

Este tipo de caso puede ser recurrente, sobretodo si consideramos que el NLHPC es utilizada por gran parte de la comunidad científica a nivel nacional. Por ejemplo, uno de los pocos gráficos que se encuentran disponibles por parte del NLHPC del uso de CPU de Guacolda para el año 2019 en la figura 5.3.2, en esta vemos un porcentaje de uso bastante alto, aunque no está al 100 %. Esto puede ser un problema, por ejemplo, si nosotros ejecutamos una tarea y necesitamos 4 núcleos para ejecutar una tarea, pero solo tenemos 2 nodos disponibles, nuestra tarea seguirá en la cola, a pesar de tener recursos disponibles. Otro caso, puede suceder es la falta de memoria RAM para nuestra tarea, aunque existan nodos libres.

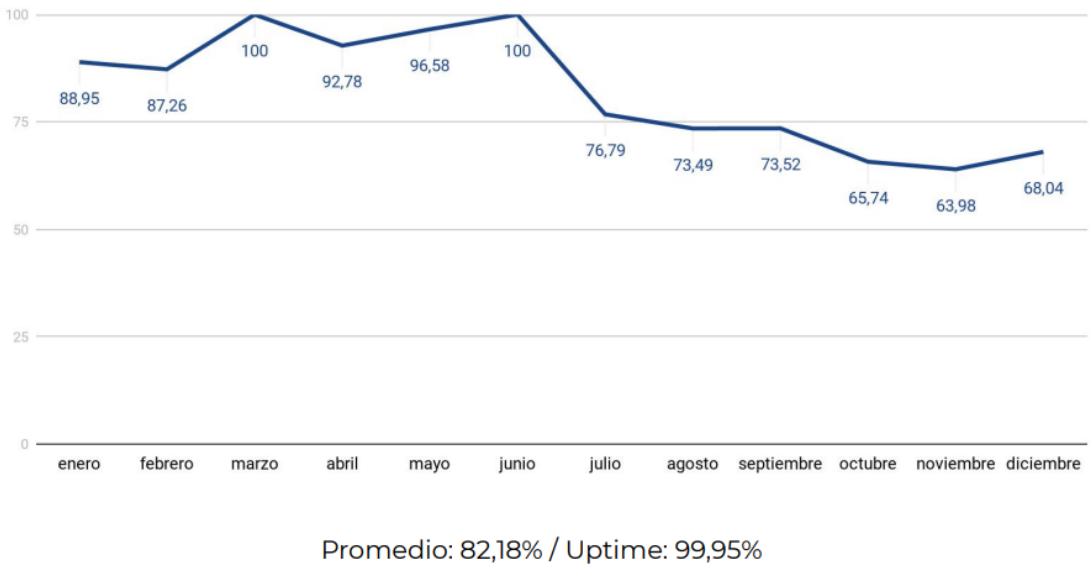


Figura 5.3.2: Representación gráfica del porcentaje de la carga de CPU de Guacolda, del 2019(NLHPC, 2022).

Bajo esta misma línea, podemos decir que los servicios de la nube a pesar de ser de pago pueden tener una respuesta mucho más rápida, comparado con el servicio que presta el NLHPC, que si consideramos que se solicita el aporte monetario, este también sería de pago. También tenemos evaluar los recursos que realmente nosotros necesitemos al momento de realizar una simulación, por lo que hacer pruebas de *speedup* y eficiencia es fundamental. Debemos considerar los recursos que necesitemos y esto dependerá de que tan compleja es nuestra simulación y también que tan rápido necesitemos los resultados.

Si nuestra elección es utilizar la nube debemos tener una noción básica o asesoramiento de los recursos de la nube, esto puede ser primordial a la hora de querer generar una rápida aplicación de nuestro modelo, porque en el caso de no tener nociones básicas, puede ser complicado implementar y entender desde el cobro hasta los diferentes servicios a utilizar.

Otro punto a considerar es el NLHPC proporciona infraestructura a diferentes cursos a nivel país y también continuamente cuenta con eventos, charlas y cursos introductorios/avanzados para usuarios del NLHPC, lo que genera un aporte para el aprendizaje para utilizar herramientas HPC, además del soporte que brindan.

Por otro lado, el mundo de la nube es bastante amplio y puede llegar a ser abrumador la cantidad de servicios que uno se puede encontrar y saber seleccionar

cual funciona de mejor manera acorde a las necesidades que se tienen como usuario. Para el caso de Azure, podemos realizar cursos en línea dependiendo del tópico que no interese, lo cual ayuda a entender más la arquitectura, infraestructura y servicios de la Nube, sumado también que existe una comunidad Global. Y en el caso de si necesita y se cuenta con los recursos económicos, se puede realizar asesoramiento, solicitando en la página de Azure los cuales cuentan con empresas asociadas en Chile disponibles para realizar este trabajo.
Por lo que la principal diferencia que podemos notar es más que nada los recursos económicos que pueden influenciar a la hora de seleccionar un servicio, sobre todo si se es estudiante y no se tienen los recursos económicos para realizar una simulación, los recursos del NLHPC pueden ser una buena alternativa.

5.4. Experiencias en implementación del modelo y futuras alternativas

En esta sección se discutirán posibles aplicaciones futuras y opciones que podrían desarrollarse o poner a prueba y ver la factibilidad ya sea en costes, desempeño o facilidad de aplicación.

5.4.1. Experiencias con Kubernetes

En los servicios de la nube también podemos encontrar otras alternativas para utilizar modelos numéricos del océano que utilizan computo de alto rendimiento, esto es a través de Kubernetes. Un ejemplo de esto es el trabajo de Jung et al. (2021), en el cual se ejecuta el modelo ROMS utilizando MPI bajo la arquitectura computacional basada en contenedores y clúster Kubernetes.
El desarrollo de contenedores puede llegar a facilitar la implementación de aplicaciones con un diseño más ligero donde se utiliza el kernel de un sistema operativo para implementar un contenedor que contendrá una aplicación ahorrando costos computacionales. Este tipo de aplicaciones pueden ser dirigidas a el desarrollo de aplicaciones y páginas web también puede ser utilizado en softwares de modelación numérica ya sea del océano o atmosférico. Bajo esta propuesta Jung et al. (2021) utilizan un clúster orquestado a través de Kubernetes que proporciona una escalabilidad y administración de contenedores sencilla para un

modelo numérico.

Por lo general para la implementación en un entorno exportable de algún software (que para este caso puede ser un modelo numérico como lo es CROCO) se implementan en MV con su imagen respectiva (un ejemplo de imágenes para MV es LiveCROCO (Sepulveda, 2022)). Ahora una MV proporciona al usuario un sistema operativo completo para trabajar, donde el usuario puede usar varios programas de aplicación alojados en ella. Pese a esto, la virtualización es como una capa adicional entre el sistema operativo y el usuario(Goldberg, 1974).

Los contenedores también utilizan la virtualización, pero diferenciándose en algunos aspectos, requieren menor cantidad de recursos y es bastante ligero. Estos contenedores comparten el mismo kernel que el *host*, donde casi todos los componentes (software y hardware) se comparten entre las aplicaciones y el sistema operativo del *host*, donde el contenedor generalmente es un paquete que incluye varias aplicaciones, como también dependencias asociadas con estas aplicaciones. Así mismo también se ve como una aplicación portátil en la computación para la nube u otro equipo, y estos caen dentro de la categoría de PaaS(Yadav et al., 2019). En la figura 5.4.1 se ejemplifica la diferencia.

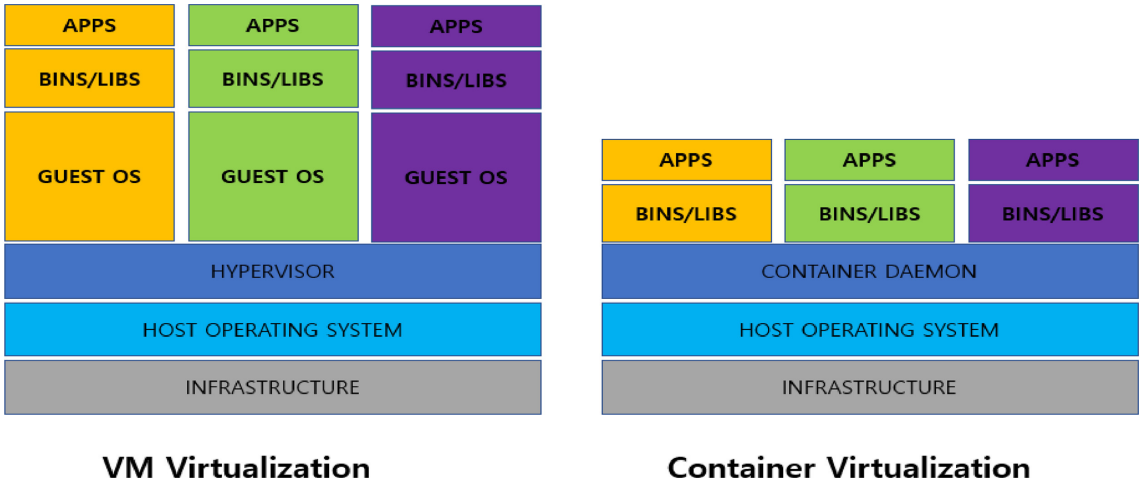


Figura 5.4.1: Comparación de de arquitectura conceptual de MV y contenedores(Jung et al., 2021).

Kubernetes es una plataforma de código abierto para la organización en contenedores que automatiza muchos de los procesos manuales involucrados en la implementación, la gestión y el ajuste de las aplicaciones que se alojan en ellos(Kubernetes-Documentación, 2021). Esta cuenta con el control y gestión de unidades básicas denominadas *pods* ubicadas en cada nodo, donde se alojan

los contenedores, y esto permite la comunicación entre cada nodo utilizando los componentes de la red, también un escalamiento dependiendo del caso. Kubernetes también administra automáticamente la detección de servicios, incorpora equilibrio de carga, realiza un seguimiento de la asignación de recursos y los escala en función del uso de la capacidad de proceso(Azure, 2023).

Toda esta administración se realiza con la herramienta kubectl que despliega los recursos. El despliegue se realiza con un archivo denominado YAML, que un lenguaje de serialización de datos legible por humanos que se usa comúnmente para archivos de configuración y en aplicaciones donde se almacenan o transmiten datos. También otro punto a destacar, que se puede desarrollar un contenedor con Docker por ejemplo y registrarla en almacenadores públicas, para luego ser desplegadas mediante Kubernetes (Kubernetes-Documentación, 2020).

Primero se necesita una imagen que contenga una imagen de un sistema operativo y paquetes necesarios para ejecutar nuestro modelo, en el caso de nosotros puede ser CROCO. Un trabajo similar es el que realiza Sepúlveda (2023)(ver detalles de un contenedor de CROCO en el Anexo A4.1).

Para el desarrollo de nuestro clúster de Kubernetes podemos utilizar diferentes proveedores de nube, que la mayoría cuenta con servicios que ofrecen clústers que ya tienen implementado Kubernetes, y estos pueden ser ajustados a los requerimientos que nosotros queramos utilizar. También tenemos la opción de nosotros poder implementar Kubernetes a un clúster privado o también generado en al Nube.

Luego de tener esto resuelto se, se necesita utilizar un tipo de controlador llamado StatefulSet, este es encargado de la distribución de contenedores a los nodos a través de unidades llamadas *pods* Kubernetes-Documentación (2022). Se utiliza StatefulSet para implementar los contenedores de modelado numérico oceánico ya que este tiene opciones para trabajar con computo de alto rendimiento. Este también contiene información de la distribución del contenedor, réplicas, número de contenedores, imagen del repositorio, puertos del contenedor, etc (Aclarar que existen otro tipo de controladores). Esto permite un escenario factible para la realización de pruebas, también un cambio intuitivo dependiendo del cambio de entorno en el que se trabaja.

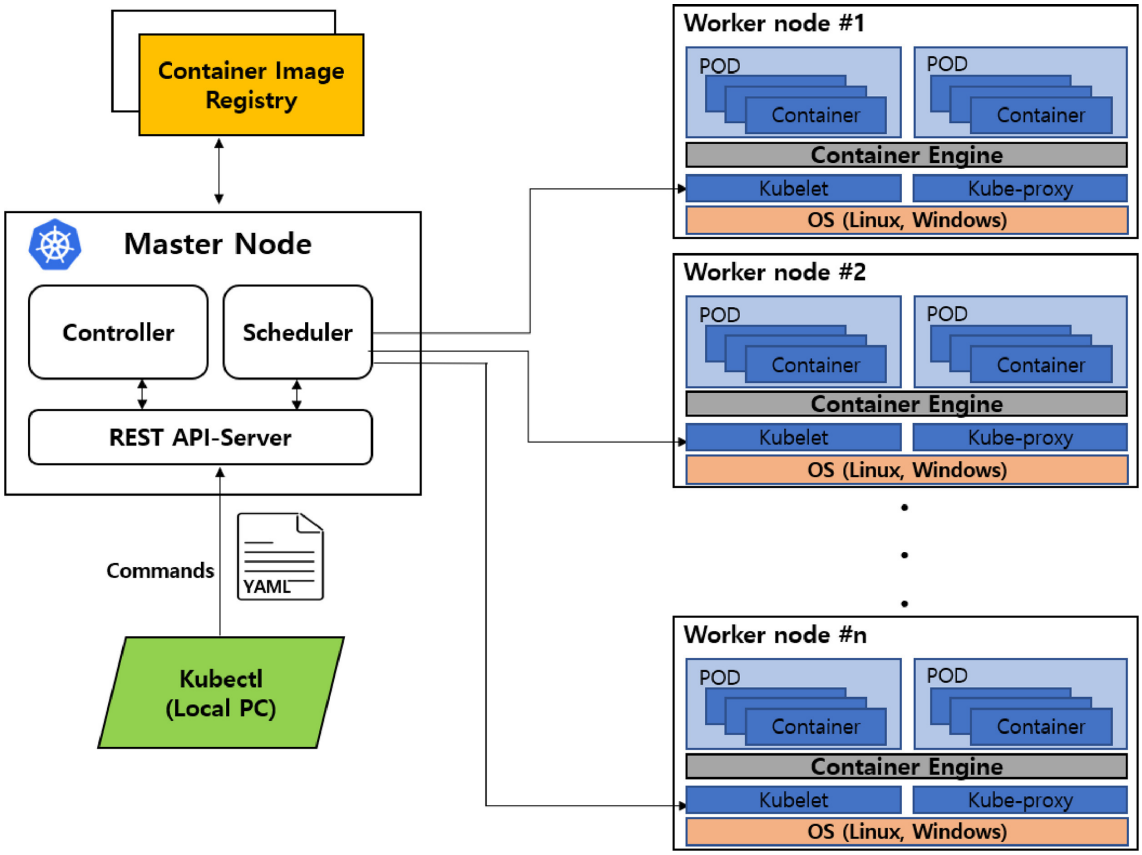


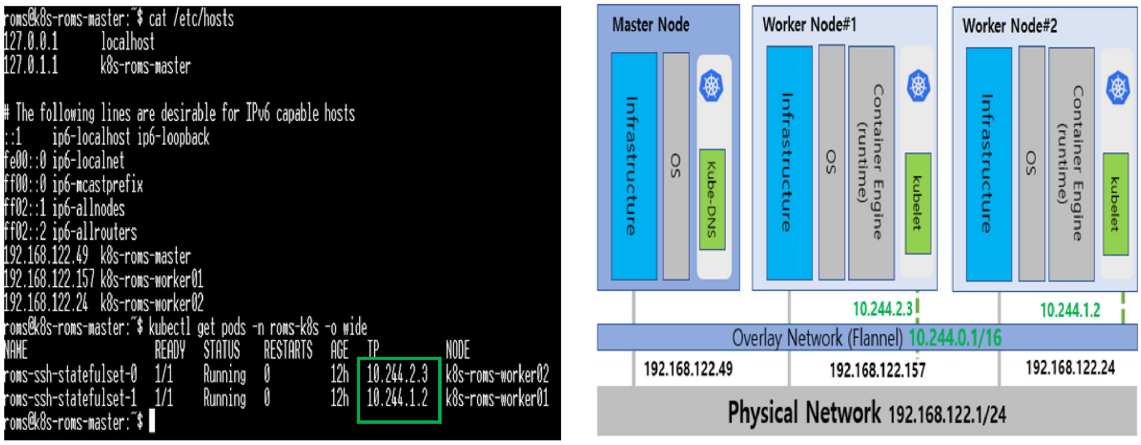
Figura 5.4.2: Ejemplo de organización y distribución de la infraestructura de un clúster de Kubernetes con un nodo Maestro y las réplicas de trabajadores(Jung et al., 2021).

5.4.2. Implementación

Primero, se realiza la construcción de un entorno en el clúster sin la necesidad de configuración de un entorno MPI adicional en el clúster de Kubernetes, aportando a la portabilidad. También tenemos que utilizar una red de superposición definido por el software que residen en la red física e implementa varios complementos de controladores de la red para distintos propósitos, para la comunicación entre *pods*. Luego declaramos el volumen de almacenamiento para el acceso a varios *pods* (que tenga permisos de lectura y escritura). Para finalmente implementar Statefulset en varios nodos de servidores, ajustando la imagen y la cantidad de pods según sea el caso (podemos ver un ejemplo de la implementación en la sección A4.2 o en el trabajo de Jung et al. (2021)).

Después de implementar los *pods* en varios nodos, se puede comprobar su estado en los clústers. También tenemos que cada pod tiene una dirección IP en la red

de superposición privada sobre la dirección IP del nodo(ver ejemplo de la Figura 5.4.4) . Esta se utiliza para ejecutar el comando *mpirun*, una vez tengamos todo listo y ejecutemos en comando una vez en el interior del contenedor.



(a) Network configuration of pods **(b) Network conceptual diagram**

Figura 5.4.3: Ejemplo de configuración de la redes para los pód y su ip, (a) configuración y obtención de los pods en un espacio de trabajo (b) Diagrama de configuración de red para los pods en la red del clúster de Kuberentes(Jung et al., 2021).

La implementación de un clúster Kubernetes es bastante rápida si es que consideramos que dentro de los proveedores de nube existe la posibilidad y capacidad de montar un clúster ya con Kubernetes(GGP, Microsoft Azure, AWS, etc). Se trata de implementar este código a modo de ensayo con GGP. El principal problema que surgió fue debido a que no todas las plataformas de la nube permiten que las réplicas de los *pods* compartan un volumen para escritura y lectura, por lo que no se puede realizar más de una réplica con StatefulSet, siendo imposible utilizar más de un nodo y MPI. Para ese tipo de pruebas se utilizó una imagen del repositorio de Docker realizada por Sepúlveda (2023), en el cual se realizó una prueba con 1 solo pod y en un solo nodo. Realizando la simulación por defecto que es el caso Benguela, aunque no se dio mayor relevancia ya que sería similar a realzar pruebas en una MV pero con capas extras. Siendo un punto importante a considerar a la hora de aplicar códigos de esta manera, tener conocimiento en tópicos de informática que se relacionan con la arquitectura de redes.

También es importante considerar la conexión entre los contenedores, donde al igual que en los clúster convecionales podemos implementar *InfiniBand* para reducir latencia como también configuración acceso remoto directo a memoria (RDMA) para reducir aun más esta. La ventaja de un clúster de Kubernetes ante un clúster tradicional (al igual que todo los recursos de la nube) es la posibilidad de escalamiento de forma más rápida.
A pesar de que este tipo de soluciones se enfocan en acercar y simplificar el trabajo de la comunidad de geocientíficos, puede llegar a ser un poco complicado, ya que se utilizan herramientas como Kubernetes que están mayoritariamente destinadas a la implementación de aplicaciones y páginas web. Para geocientíficos sin conocimiento en arquitectura de redes o servicios en la nube, implementar MV es una opción más accesible como primeros pasos.

5.4.3. Clúster en Azure

La generación de un clúster en Azure o cualquier proveedor de la nube puede llegar a ser un poco complicado sobretodo si no tenemos conocimientos específicos en el mundo de la informática y arquitectura de redes (ya sea local o en la nube). Para esto Azure contiene alguno de sus servicios como Azure CycleCloud, que permite orquestar y administrar clúster para aplicaciones HPC. También la misma comunidad de Azure ha desarrollado plantillas que permiten el desarrollo de aplicaciones HPC, tales como Slurm, PBSPro, LSF, Grid Engine y HT-Condor. CycleCloud es el producto hermana para Azure Batch, que nos permite también ejecutar aplicaciones de forma paralela.
Una forma más simple también podemos generar clúster para la ejecución de aplicaciones con MPI, uno de los trabajos introductorios para para esto es de Lockwood (2022), que propone un ejercicio para la generación de un clúster de prueba(se puede revisar parte de este en Anexo A7). Para la implementación de este propone utilizar un entorno PPG (descrito en la metodología) para la disminución de la latencia. También utiliza la conectividad entre *host* a través de SSH con las Keys generadas en cada nodo. Ocupa la herramienta **clush**, esta es un programa para ejecutar comandos en paralelo en un clúster y para recopilar resultados de estos, cumpliendo la utilidad de copiar archivos y carpetas en los diferentes nodos a seleccionar, este también nos sirve para la creación de un

sistema de red de archivos compartidos utilizando NFS, aunque esto puede no ser escalable(Thiell, 2023).
Este tipo de trabajo puede ser un buen ejercicio de aprendizaje para entender como generar un clúster y ejecutar ejemplo de código con MPI. Aunque también hay que entender que para CROCO y simulaciones más realistas se necesite una implementación más avanzada y más automatizada. En ese sentido una la creación de una MV es un nivel bastante sencillo de aplicar, pero la creación de un clúster puede generar complicaciones, sobretodo si queremos hacerlo manualmente.

5.4.4. Implementación de modelo IooS

Basado en una Sandbox , el modelo Coastal Modeling Cloud Sandbox proporciona un entorno virtual aislado en la nube donde se pueden ejecutar modelos costeros regionales. Este aprovecha entornos HPC de la nube, administrados a través de Terraform y el proveedor de servicios de la nube AWS.

5.4.4.1. ¿Qué es Terraform?

Terraform es una herramienta de orquestación de código abierto que es desarrollado por la empresa HashiCorp. Esta herramienta de codificación declarativa permite a los desarrolladores la configuración para describir una infraestructura desea sea en la nube(Amazon Web Services (AWS), Azure, Google Cloud Platform (GCP), Kubernetes, Helm, GitHub, Splunk, DataDog y muchos más) o a nivel local, para ejecutar una aplicación, esta configuración se llama HCL(*HashiCorp Configuration Language*). Entonces como finalidad tiene el poder aprovisionar y administrar infraestructura de manera más eficiente y segura, y en este caso la podemos denominar infraestructura como código(Terraform, 2023).
Bajo este contexto, entendemos si es que necesitamos una buena administración, y queremos tener un buen flujo de trabajo, orquestando de buena forma los recursos en la nube para el desarrollo de un modelo que requiera HPC, Terraform es la herramienta ideal. Este tipo de herramientas es conocida como insfraestructura como código.

5.4.4.2. Instalación de Coastal Modeling CloudSandbox

Para la implementación del Coastal Modeling CloudSandbox se utiliza AWS. Para la conexión a las maquinas se utilizan las llaves SSH de AWS y todos los archivos de configuración se encuentran presentes en el repositorio Github del proyecto. Para la implementación de los modelos se ocupa el código de fuente NOSFS 3.5 y que pretenden integrar modelos como ROMS,SCHIMS,ADCIRC y FVCOM. Instalado todo lo necesario solo nos queda utilizar nos quedaría por implementar la API CloudFlow, donde tenemos que adaptar nuestros requisitos para la utilización de esta API a lo solicitado de recursos de AWS y configuraciones del modelo. También se tienen herramientas para el postprocesamiento de las salidas. Esta API esta distribuida de la siguiente forma:

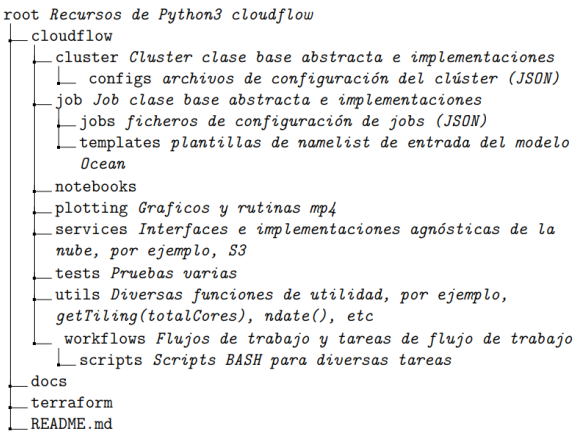


Figura 5.4.4: Distribución de los recursos de *Coastal Modeling CloudSandbox* en un digrama de árbol.

Ahora, considerando que este modelo está en desarrollo y que también que se generará una comunidad al rededor de este proyecto, puede ser una buena alternativa para el desarrollo de modelos en la nube, integrando la infraestructura bajo la aplicación de herramientas como Terraform, lo que también puede ayudar a mejorar flujos de trabajos, sumado también que contiene herramienta para el post procesamiento de los resultados. Se puede considerar que es una buena oportunidad para implementar el modelo numérico del océano como lo es CROCO. Ayudando a generar la administración de recursos más avanzadas, en comparación a solo solicitar una MV. Sin embargo, este tiene un grado mayor de complejidad para un geocientífico que recién se introduce a los servicios de la nube.

Capítulo 6

Conclusión

El trabajo desarrollado nos hace entender de la importancia de los recursos informáticos y como estos ayudan al avance y procesos de la ciencia y modelación, esto no solo como máquinas o supercomputadores, sino también de la difusión de resultados, recogida de datos, almacenamiento y accesibilidad del conocimiento, conocido como *e-science*.

El acceso, mercantilización, acceso computadores, soluciones para obtener a un mayor poder computacional, clústers, computación en malla, el computo de alto rendimiento y la paralelización de programas. Todas soluciones permiten obtener un rendimiento más óptimo para nuestros modelos, cada día más complejos.

La nube como una nueva alternativa para acceso de recursos informáticos, suma a otras soluciones como son los clúster, virtualización, computo de alto rendimiento y variadas herramientas suman una variada gama de herramientas que pueden ser aprovechadas por el mundo de la modelación. Sin la necesidad de adquirir una máquina o supercomputador ni la mantención de estos, siendo un entorno mucho más flexible y escalable.

Se evidencia que para la investigación de aplicación de modelos numéricos del océano en la nube, no es tan amplia como si lo existe en el área de modelación numérica de la atmósfera, principalmente asociada al pronóstico. A pesar de esta brecha que existe en la investigación, se puede llegar a adaptar de manera generalizada.

La implementación de una MV es bastante rápida cuando se tiene interiorizado los conceptos y se utilizan herramientas como Azure CLI. Lo que puede llegar a complicar la implementación es no entender los conceptos de informática o redes

siendo nuevo en el mundo de la nube. Aunque existe gran cantidad material para este caso, lo que permitió para este trabajo implementar de buena forma la MV y realizar nuestra simulación realista, cumpliendo con el objetivo específico 1.
Ahora, si se quiere realizar una simulación es importante realizar pruebas de rendimiento y asesorarse en el correcto uso de la plataforma que se desea ocupar. Para el caso de crear una MV en la nube para implementar CROCO solo es necesario lo que se detalla en la guía de inicio rápido de los resultados.
En cuanto a las pruebas realizadas en este trabajo no se genera un gran gasto, siendo de 0.39 USD solo para las pruebas. Esto debido a que se realiza solo para un solo día, no se aplica ningún otro modulo, también no se utiliza MV con una gran cantidad de vCPU y no realiza transferencia de datos para descarga de resultado. Todo esto podría haber aumentado el precio por hora del uso de las MVs.
A pesar de que se realizan pruebas solo con 4 vCPU para la simulación, con la paralelización con OpenMP se logra visualizar que la simulación esta al borde del límite de eficiencia. Respecto a las distribuciones de paralelización del dominio, se obtuvo que en la dirección latitudinal (eta) se obtiene un menor tiempo y mayor eficiencia, esto con respecto a la dirección meridional (xi) o al caso isométrico. También notar que se realizó una grabación horaria de las simulaciones, esto aumentó el tiempo de cálculo generando una eficiencia mucho menor, sin embargo, se proponen soluciones para mejorar la eficiencia en este aspecto.
A pesar de que se seleccionó una MV de la serie Fs_v2 que para la descripción óptima para operaciones de punto flotante y procesos optimizados, se compara con los resultados de otras pruebas realizadas (MV D5as_v4) y se estima que para la misma simulación se reduce en tiempo de cálculo y coste con la segunda MV, lo que llama la atención, recomendando realizar pruebas a futuro en otras series de MV. Con esto último se cumple el objetivo específico 2.
Se discute el uso de NLHPC y los requerimientos que este pide, concluyendo que al pedir un aporte tampoco es un servicio totalmente gratuito ya que forma parte de la evaluación de admisión, sumado también otros problemas que se generan por la gran demanda de los recursos de este clúster, pero no deja de ser una opción para estudiantes o investigadores. La nube y MV pueden ser una buena solución, para acceder a recursos de una forma más rápida sin procesos de admisión, flexible y escalable, pero esto dependerá si se cuenta con recursos económicos.
Se propone y compara con otros servicios que pueden ofrecer la nube. Por ejemplo, formar un clúster en Azure con servicios que el mismo proveedor, concluyendo que

puede ser complejo de aplicar ya que se requiere conocimiento de arquitectura de redes. También se habla de la aplicación de Kubernetes, que a pesar de una buena implementación, replicable y con posibilidad de utilizar recursos para computo de alto rendimiento, también requiere de conocimiento de arquitectura de redes y puede ser un reto aprender conceptos nuevos sobre la implementación de recursos a través de esta API.

La alternativa que puede llegar a tener un futuro prometedor para la implementación de modelos numéricos del océano, es *Coastal Modeling CloudSandbox*, que utiliza Terraform y también una propia API, esto simplifica bastante la aplicación de modelos en la nube y da la oportunidad a desarrollar incluso clúster HPC y diferentes flujos de trabajo, pudiendo aplicar modelos y también acoplarlos. Próximamente se desarrollará un foro entorno a este modelo, lo cual será importante para la interacción entre usuarios, soporte y resolución de problemas, creación de conocimiento, retroalimentación e incluso incorporar otros modelos como puede ser CROCO.

Con la discusión de diferentes alternativas para computo de alto rendimiento en la discusión, se cumple con el objetivo específico 3, proponiendo futuras alternativas de implementación y herramientas que prometen un futuro para el desarrollo de los modelos numéricos del océano o atmosféricos en la Nube.

Este trabajo puede ser considerado como un acercamiento al entorno de la nube, donde no solo se pueden correr modelos sino también almacenar datos y generar aplicaciones de páginas WEB que permitan un acceso a datos, ya sean observacionales, reanálisis o de modelos, como se da el caso de Fatland et al. (2016). Donde se ve que los recursos pueden ser implementados de diferentes formas para el desarrollo de la ciencia en general, esto último cumpliendo con el objetivo general de este trabajo, viendo diferentes implementaciones del modelo numérico del océano en no solo con un servicio de nube sino, entendiendo que se pueden utilizar diferentes servicios y proveedores acordes a nuestras necesidades y el desarrollo de herramientas.

Finalmente podemos para concluir con una pregunta Zhuang et al. (2019) ¿Cómo se pueden financiar este tipo de herramientas? Por mi parte, creo que pueden ser financiados por de proyectos del Estado o fondos públicos para investigadores. También pueden ser una buena herramienta para universidades que requieran poder o infraestructura para sus estudiantes, generando un entorno flexible, con accesibilidad y capacidad de escalamiento según la necesidad sus necesidades.

Bibliografía

Abernathey, R. P., Augspurger, T., Banihirwe, A., Blackmon-Luca, C. C., Crone, T. J., Gentemann, C. L., Hamman, J. J., Henderson, N., Lepore, C., McCaie, T. A., Robinson, N. H., and Signell, R. P. (2021). Cloud-native repositories for big scientific data. *Computing in Science Engineering*, 23(2):26–35.

Ahmed, O. M., Haji, L. M., Shukur, H. M., Zebari, R. R., Abas, S. M., and Sadeeq, M. A. (2020). Comparison among cloud technologies and cloud performance. *Journal of Applied Science and Technology Trends*, 1:40–47.

Aljamal, R., El-Mousa, A., and Jubair, F. (2018). A comparative review of high-performance computing major cloud service providers. In *2018 9th International Conference on Information and Communication Systems (ICICS)*, pages 181–186. IEEE.

Artal, O., Pizarro, O., and Sepúlveda, H. H. (2019). The impact of spring-neap tidal-stream cycles in tidal energy assessments in the chilean inland sea. *Renewable Energy*, 139:496–506.

Auclair, F., Benshila, R., Bordois, L., Boutet, M., Brémond, M., Caillaud, M., Cambon, G., Capet, X., Debreu, L., Ducousso, N., Dufois, F., Dumas, F., Ethé, C., Gula, J., Hourdin, C., Illig, S., Jullien, S., Corre, M. L., Gac, S. L., Gentil, S. L., Lemarié, F., Marchesiello, P., Mazoyer, C., Morvan, G., Nguyen, C., Penven, P., Person, R., Pianezze, J., Pous, S., Renault, L., Roblou, L., Sepulveda, A., and Theetten, S. (2022). Coastal and regional ocean community model.

Azure (2023). ¿qué es kubernetes? https://azure.microsoft.com/es-mx/resources/cloud-computing-dictionary/what-is-kubernetes/#overview (última vez revisado el 01/06/2023).

Bohle, S. (2013). Bohle, shannon. (2013, 12 june). "what is e-science and how should it be managed?"scientific and medical libraries. scilogs. nature and spektrum der wissenshaft.

Bonet Esteban, E. V. (2014). Servicios de acceso remoto ii: Ssh. *Apunte de curso*.

Cao, W., Guo, Y., Zhang, H., Sun, M., and Yuan, M. (2021). Application of container technology in numerical ocean model: A kind of high-performance

roms containerized architecture. In *Journal of Physics: Conference Series*, volume 1961, page 012046. IOP Publishing.

Chen, X., Huang, X., Jiao, C., Flanner, M. G., Raeker, T., and Palen, B. (2017). Running climate model on a commercial cloud computing environment: A case study using community earth system model (cesm) on amazon aws. *Computers Geosciences*, 98:21–25.

Chui, T. C., Siuta, D., West, G., Modzelewski, H., Schigas, R., and Stull, R. (2019). On producing reliable and affordable numerical weather forecasts on public cloud-computing infrastructure. *Journal of Atmospheric and Oceanic Technology*, 36(3):491–509.

Clarke, L., Glendinning, I., and Hempel, R. (1994). The mpi message passing interface standard. In *Programming environments for massively parallel distributed systems*, pages 213–218. Springer.

Contreras, M., Pizarro, O., Dewitte, B., Sepulveda, H. H., and Renault, L. (2019). Subsurface mesoscale eddy generation in the ocean off central chile. *Journal of Geophysical Research: Oceans*, 124(8):5700–5722.

CROCO (2020). Documentación de CROCO, paralelización. https://croco-ocean.gitlabpages.inria.fr/croco_doc/model/model.parallel.html(última vez revisado el 22/11/2022).

CROCO (2023a). Documentation. https://www.croco-ocean.org/documentation-2/ (última vez revisado el 10/06/2023).

CROCO (2023b). System requirements. https://croco-ocean.gitlabpages.inria.fr/croco_doc/tutos/tutos.00.env.html (última vez revisado el 10/06/2023).

Dagum, L. and Menon, R. (1998). Openmp: an industry standard api for shared-memory programming. *Computational Science & Engineering, IEEE*, 5(1):46–55.

Del Vecchio, J. F., Paternina, F. J., and Miranda, C. H. (2015). La computación en la nube: un modelo para el desarrollo de las empresas. *Prospectiva*, 13(2):81–87.

Fatland, R., MacCready, P., and Oscar, N. (2016). Liveocean. In *Cloud Computing in Ocean and Atmospheric Sciences*, pages 277–296. Elsevier.

Flores Gil, A. (2004). Evaluación del rendimiento. Universidad de Murcia.

Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud computing and grid computing 360-degree compared. In *2008 grid computing environments workshop*, pages 1–10. Ieee.

Garvey, C. (2020). Run wrf v4 on azure hpc virtual machines. https://techcommunity.microsoft.com/t5/azure-high-performance-computing/run-wrf-v4-on-azure-hpc-virtual-machines/ba-p/1131097 (última vez revisado el 26/04/2023).

Gent, P. R., Danabasoglu, G., Donner, L. J., Holland, M. M., Hunke, E. C., Jayne, S. R., Lawrence, D. M., Neale, R. B., Rasch, P. J., Vertenstein, M., et al. (2011). The community climate system model version 4. *Journal of climate*, 24(19):4973–4991.

Gill, S. S. and Buyya, R. (2019). Sustainable cloud computing realization for different applications: a manifesto. *Digital Business: Business Algorithms, Cloud Computing and Data Engineering*, pages 95–117.

Goldberg, R. P. (1974). Survey of virtual machine research. *Computer*, 7(6):34–45.

Herrera, J., Cornejo, P., Sepúlveda, H. H., Artal, O., and Quiñones, R. A. (2018). A novel approach to assess the hydrodynamic effects of a salmon farm in a patagonian channel: coupling between regional ocean modeling and high resolution les simulation. *Aquaculture*, 495:115–129.

IBM (2021). Hpc - computación de alto rendimiento. https://www.ibm.com/mx-es/topics/hpc. Consultado el 6 de abril de 2023.

Integrated Ocean Observing System (2023). Coastal modeling cloud sandbox.

Jullien, S., Caillaud, M., Benshila, R., Bordois, L., Cambon, G., Dufois, F., Gula, J., Le Corre, M., Le Gentil, S., Lemarié, F., et al. (2022). Croco tutorials.

Jung, K., Cho, Y.-K., and Tak, Y.-J. (2020). Install guide and deployment codes for Jung et al., Simulation Modelling Practice and Theory: Containers and Orchestration of the Numerical Ocean Model for Computational Reproducibility and Portability in Public and Private clouds: Application of ROMS 3.6.

Jung, K., Cho, Y.-K., and Tak, Y.-J. (2021). Containers and orchestration of numerical ocean model for computational reproducibility and portability in public and private clouds: Application of roms 3.6. *Simulation Modelling Practice and Theory*, 109:102305.

Kubernetes-Documentación (2020). Entender los objetos de kubernetes. https://kubernetes.io/es/docs/concepts/overview/working-with-objects/kubernetes-objects/ (última vez revisado el 01/06/2023).

Kubernetes-Documentación (2021). ¿qué es kubernetes? https://kubernetes.io/es/docs/concepts/overview/what-is-kubernetes/ (última vez revisado el 01/06/2023).

Kubernetes-Documentación (2022). Controller. https://kubernetes.io/docs/concepts/architecture/controller/ (última vez revisado el 30/05/2023).

Learn-Microsoft (2023). https://learn.microsoft.com/es-es/azure/virtual-machines/snapshot-copy-managed-disk?tabs=portal. https://learn.microsoft.com/es-es/azure/virtual-machines/snapshot-copy-managed-disk?tabs=portal (última vez revisado el 07/06/2023).

Li, J., Liu, K., and Huang, Q. (2016). Utilizing cloud computing to support scalable atmospheric modeling: A case study of cloud-enabled modele. In *Cloud Computing in Ocean and Atmospheric Sciences*, pages 347–364. Elsevier.

Lockwood, G. K. (2022). Quick mpi cluster setup on azure. https://www.glennklockwood.com/cloud/mpi-cluster.html (última vez revisado el 26/05/2023).

Mell, P., Grance, T., et al. (2011). The nist definition of cloud computing.

Microsoft-Learn (2023a). Introducción a los discos administrados de azure. https://learn.microsoft.com/es-es/azure/virtual-machines/managed-disks-overview (última vez revisado el 10/06/2023).

Microsoft-Learn (2023b). Tamaños de máquina virtual de uso general. https://learn.microsoft.com/es-es/azure/virtual-machines/sizes-general (última vez revisado el 30/05/2023).

Molthan, A. L., Case, J. L., Venner, J., Schroeder, R., Checchi, M. R., Zavodsky, B. T., Limaye, A., and O'Brien, R. G. (2015). Clouds in the cloud: weather forecasts and applications within cloud computing environments. *Bulletin of the American Meteorological Society*, 96(8):1369–1379.

Netto, M. A., Calheiros, R. N., Rodrigues, E. R., Cunha, R. L., and Buyya, R. (2018). Hpc cloud for scientific and business applications: taxonomy, vision, and research challenges. *ACM Computing Surveys (CSUR)*, 51(1):1–29.

NLHPC (2020). Introducción al uso de la infraestructura del nlhpc.

NLHPC (2022). Nlhpc, national laboratory for high performance computing. https://www.nlhpc.cl/acerca/ (última vez revisado el 13/04/2022.

Nlhpc-Web (2022). Introducción al uso de la infraestructura del nlhpc. https://www.nlhpc.cl/infraestructura/ (última vez revisado el 22/11/2022).

Nlhpc-Wiki (2022). Estudio de eficiencia. https://wiki.nlhpc.cl/Estudio_de_Eficiencia (última vez revisado el 22/11/2022).

Penven, P., Cambon, G., Marchesiello, P., Sepulveda, A., Benshila, R., Illig, S., Jullien, S., Corre, M. L., Gentil, S. L., and Morvan, G. (2022). Croco tools.

Rashid, A. and Chaturvedi, A. (2019). Cloud computing characteristics and services: a brief review. *International Journal of Computer Sciences and Engineering*, 7(2):421–426.

Reche, P., Artal, O., Pinilla, E., Ruiz, C., Venegas, O., Arriagada, A., and Falvey, M. (2021). Chonos: Oceanographic information website for chilean patagonia. *Ocean & Coastal Management*, 208:105634.

Riha, S. (2017). The roms ocean model on aws.

Sepulveda, A. (2022). Livecroco$_v$1,2*beta*.

Sepúlveda, A. (2023). Docker image to run the coastal and regional ocean community model - croco. https://github.com/AndresSepulveda/docker-croco-public (última vez revisado el 29/05/2023).

Shchepetkin, A. F. and McWilliams, J. C. (2005). The regional oceanic modeling system (roms): a split-explicit, free-surface, topography-following-coordinate oceanic model. *Ocean modelling*, 9(4):347–404.

Siuta, D., West, G., Modzelewski, H., Schigas, R., and Stull, R. (2016). Viability of cloud computing for real-time numerical weather prediction. *Weather and Forecasting*, 31(6):1985–1996.

Terraform (2023). What is terraform? https://developer.hashicorp.com/terraform/intro (última vez revisado el 10/06/2023).

Thiell, S. (2023). clush. https://clustershell.readthedocs.io/en/latest/tools/clush.html (última vez revisado el 01/06/2023).

Vance, T. C., Merati, N., Yang, C., and Yuan, M. (2016). *Cloud computing for ocean and atmospheric science*. IEEE.

Wankhede, P., Talati, M., and Chinchamalatpure, R. (2020). Comparative study of cloud platforms-microsoft azure, google cloud platform and amazon ec2. *J. Res. Eng. Appl. Sci*, 5(02):60–64.

Yadav, A. K., Garg, M., and Ritika (2019). Docker containers versus virtual machine-based virtualization. In *Emerging Technologies in Data Mining and Information Security: Proceedings of IEMIS 2018, Volume 3*, pages 141–150. Springer.

Zhuang, J., Jacob, D. J., Gaya, J. F., Yantosca, R. M., Lundgren, E. W., Sulprizio, M. P., and Eastham, S. D. (2019). Enabling immediate access to earth science models through cloud computing: Application to the geos-chem model. *Bulletin of the American Meteorological Society*, 100(10):1943–1960.

Anexo

A1. Máquinas Virtuales de Azure

Tipo	Tamaños	Descripción
Uso general	B, Dsv3, Dv3, Dasv4, Dav4, DSv2, Dv2, Av2, DC, DCv2, Dpdsv5, Dpldsv5, Dpsv5, Dplsv5, Dv4, Dsv4, Ddv4, Ddsv4, Dv5, Dsv5, Ddv5, Ddsv5, Dasv5, Dadsv5	Uso equilibrado de la CPU en proporción de memoria. Ideal para desarrollo y pruebas, bases de datos pequeñas o medianas, y servidores web de tráfico bajo o medio.
Optimizada para proceso	F, Fs, Fsv2, FX	Uso elevado de la CPU en proporción de memoria. Bueno para servidores web de tráfico medio, aplicaciones de red, procesos por lotes y servidores de aplicaciones.
Memoria optimizada	Esv3, Ev3, Easv4, Eav4, Epdsv5, Epsv5, Ev4, Esv4, Edv4, Edsv4, Ev5, Esv5, Edv5, Edsv5, Easv5, Eadsv5, Mv2, M, DSv2, Dv2	Memoria alta en proporción de CPU. Excelente para servidores de bases de datos relacionales, memorias caché de capacidad media o grande y análisis en memoria.
Almacenamiento optimizado	Lsv2, Lsv3, Lasv3	Alto rendimiento de disco y de E/S ideales para macrodatos, bases de datos SQL y NoSQL, almacenamiento de datos y bases de datos transaccionales grandes.
GPU	NC, NCv2, NCv3, NCasT4_v3, ND, NDv2, NV, NVv3, NVv4, NDasrA100_v4, NDm_A100_v4	Máquinas virtuales especializadas específicas para la representación de gráficos pesados y la edición de vídeo, así como para el entrenamiento e inferencia de modelos (ND) con aprendizaje profundo. Están disponibles con uno o varios GPU.
Proceso de alto rendimiento	HB, HBv2, HBv3, HBv4, HC, HX	máquinas virtuales de CPU más rápidas y eficaces con interfaces de red de alto rendimiento opcionales (RDMA).

Cuadro A1.1: Clasificación de máquinas virtuales según su tipo de uso, tamaños y pequeña descripción (Microsoft-Learn, 2023b)

A2. Creación de cuenta de prueba gratis Microsft Azure

Primero ingresamos a la página principal de Azure Microsoft (https://azure.microsoft.com/es-es/). Donde tenemos la opción de ingresar desde cero o utilizar nuestro correo Udec(Ej: xxxxx@udec.cl), como se muestra en la siguiente imagen:

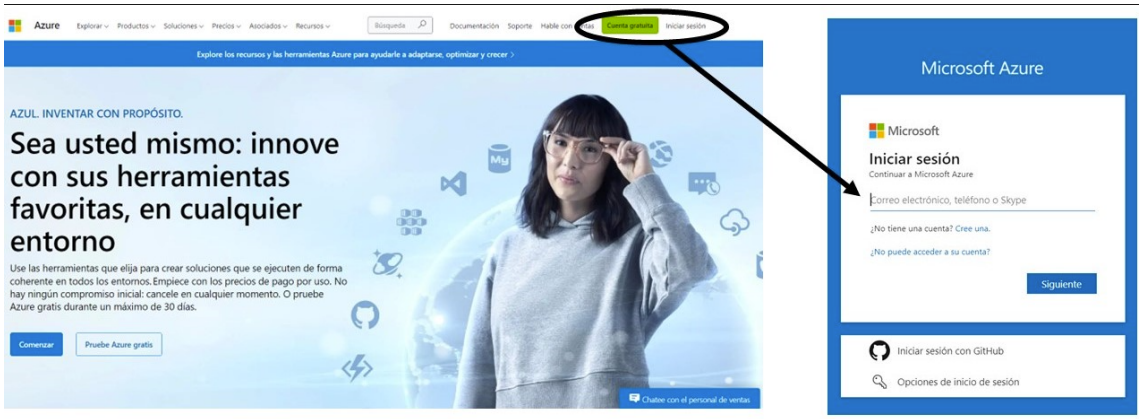


Figura A2.1: Sitio web de Microsoft Azure

Luego de esto usted entrará en la página de Azure, donde como se ve en la figura A5.4, el paso 1 es realizar un guía de la página(se recomienda realizar ya que detalla el entorno de trabajo). Para el paso 2 se selecciona la prueba gratuita que se realizará, esta contiene lo siguiente:

1. 200 USD de crédito para los primeros 30 días después de registrarse
2. Servicios populares gratuitos por 12 meses
3. 40 servicios por siempre gratis (más información en https://azure.microsoft.com/es-es/free/free-account-faq/)

El paso 3 es luego de dar inicio la prueba gratis, para luego el paso final de registro, donde se detalla los beneficios y se solicitan algunos datos. Para el caso del correo institucional no se requiere una tarjeta de crédito, en caso de no usarlo, obligatoriamente esta es requerida. En resumen, como la siguiente imagen:

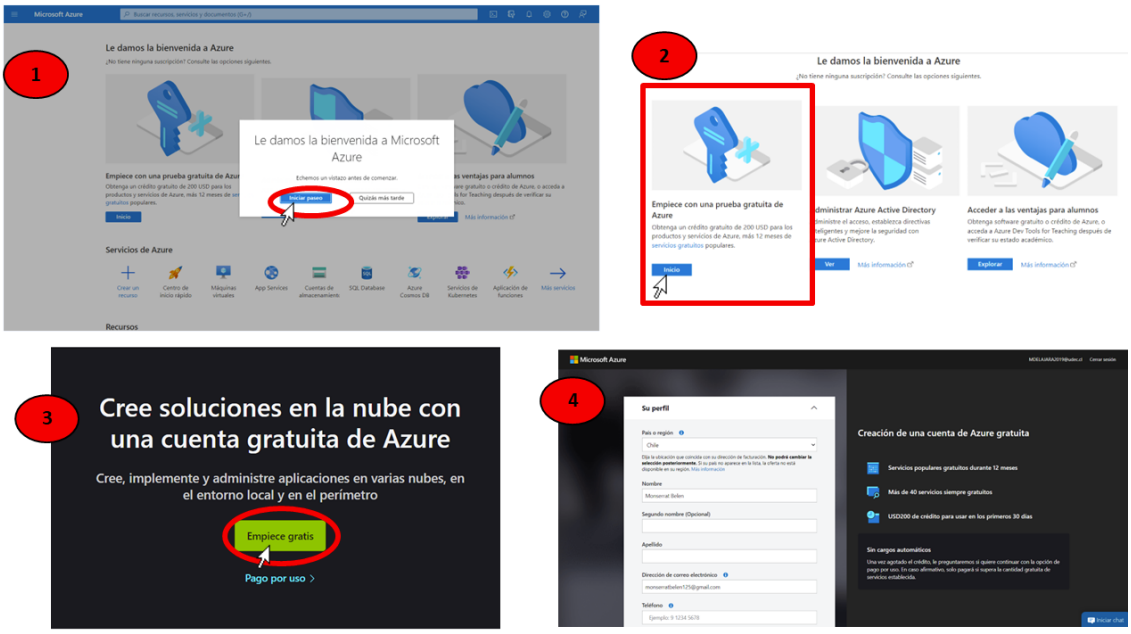


Figura A2.2: Pasos para la creación de cuenta en Microsoft

Finalmente usted queda registrado en Microsoft Azure.

A3. NLHPC

NLHPC cuenta con variados recursos disponibles a los diferentes usuarios que cuentan con el acceso a sus clústers, como se ve en la tabla A3.2.

Equipos	Leftraru - Clúster HPE (2014)	Guacolda - Clúster Dell (2019)
Cores	2640	2596
RAM	6308 GB	16.235 GB
Nodos	128 nodos HPE ProLiant Sl230s 4 nodos HPE ProLiant SL250s Gen8	48 nodos Dell Power Edge C6420 9 nodos Dell PowerEdge R640 2 nodos Dell PowerEdge R740 con 2 GPu NVDIA Tesla V100
GPU cores	No contiene	20.480
TFlops	70 TFlops (rendimiento teórico)	196 TFlops(rendimiento teórico)
CPUs	Intel Xeon E5-2660 v2	Intel Xeon Gold 6152

Cuadro A3.1: Comparación de ambos clústers que posee el Laboratorio Nacional de Computación de Alto Rendimiento actualmente(Nlhpc-Web, 2022).

Aunque la cifra más general del total de los recursos informáticos, considerando los dos supercomputadores y actualizada el 2019(Nlhpc-Web, 2022) :

- 5236 CPU cores
- 20480 GPU cores
- 266 TFlops
- 4 PB de almacenamiento
- 22 TB RAM

Este juega una papel relevante ya que las cuentas van dirigidas a la investigación, donde se puede realizar una postulación y de quedar aprobado tendrá un usuario con la duración de un año, teniendo cuentas de iniciación, con un tope de recursos (88 núcleos de cómputo, 100 gb de espacio de almacenamiento y 50.000 horas de cómputo), y también cuentas de investigación destinadas a una gran cantidad de recursos de los investigadores donde se requieren diferentes datos y luego una prueba de escalamiento de la aplicación.
El sistema del NLHPC sigue el siguiente modelo:

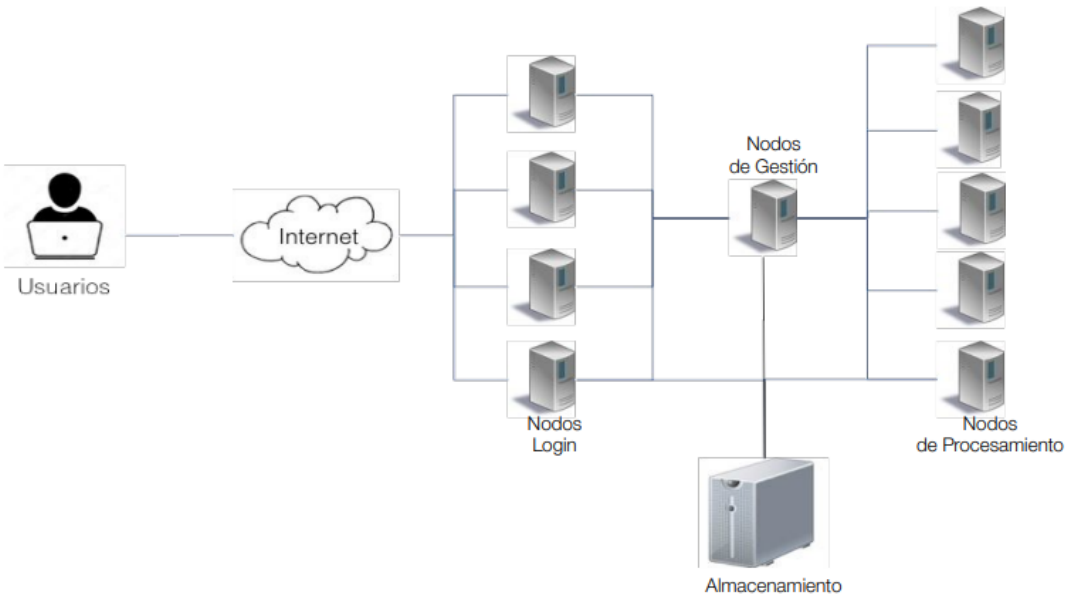


Figura A3.1: Distribución de los nodos del centro NLHPC (NLHPC, 2020).

El usuario se conecta de manera remota a algún nodo de acceso (balanceados por carga) a través de protocolo ssh a los servidores, de esta manera en los nodos login se visualiza el programa, se ve que está a la espera para ejecutar, luego de esto se lanza, va un sistema gestor, donde este tiene la capacidad de gestar los diferentes recursos que están disponibles y va asignando los trabajos.
De forma más concreta el NLHPC usa un sistema de programación de trabajos y gestión de clúster de código abierto llamado Slurm, que es tolerante a fallas y escalable para clúster Linux.
Slurm tiene tres funciones claves. Primero, asigna de acceso exclusivo y/o no exclusivo a los nodos de cómputo a los usuarios en el periodo de tiempo que se ejecuta un trabajo. Segundo, proporciona un marco para iniciar, ejecutar y monitorear el trabajo (para trabajos paralelos en nodos asignados). Tercero, gestiona una cola para trabajos solicitados que quedan pendiente.
Dentro de la solicitud hay ciertas características claves: el conjunto de recursos solicitados(numero de nodos, CPUs o núcleos), cantidad de memoria y tiempo necesario para que las tareas del usuario completen su trabajo; una partición de nodo solicitada; la calidad de servicio solicitado; el tipo de cuenta, ya que esta ultima tiene designada cierta cantidad de recursos. El trabajo requerido es enviado a una partición particular(donde podemos ver el/los nodo/s) y bajo la cuenta que tenemos(Nlhpc-Wiki, 2022).

Dentro del desarrollo de este trabajo se implementaron pruebas con *OpenMP*. Se planteó en un principio comparar el rendimiento de NLHPC pero, generando los archivos de simulación necesarios, para esto se utilizó un nodo de la partición slims con una cantidad de 20 CPU, memoria de almacenamiento de 46 GB y donde se utilizan 2000 MB de memoria RAM por CPU. Dentro de la implementación se presentaron diferentes problemas, ya que no siempre se designaba el mismo nodo, se realizaron 3 intentos con diferentes distribuciones, lo que resultó con tiempo diferentes.

Distribución	Intento 1[s]	Intento 2[s]	Intento 3[s]
1×1	367	618	379
2×2	521	583	412
4×4	694	793	402
4×4	851	766	441
2×8	578	625	420

Cuadro A3.2: Pruebas realizadas en NLHPC para simulación de 1 día.

Se implementa el siguiente código:

```
#!/bin/bash
#SBATCH --job-name=ejemplo
#SBATCH --partition=slims
#SBATCH -n 1
#SBATCH --ntask-per-node=1
#SBATCH --reservation=croco
#SBATCH --mem=2000
ml purge
ml intel/2019b
OMP_NUM_THREADS=1
start=$(date +%s.%N)
srun ./croco croco.in
end=$(date +%s.%N)
tiempo =$(echo "$end - $start" | bc)
echo $tiempo
```

La diferencia de tiempo se cree que fue debido a una posible diferencia de rendimiento en los diferentes nodos.

A4. Docker y Kubernetes

A4.1. Construcción de Imagen de Docker

Para la construcción de nuestra imagen de Docker para implementarla en un contenedor es necesario tener nuestro dockerfile, de esta forma podremos implementarla en algún repositorio público o generarla en nuestra máquina o clúster de Kubernetes. Un ejemplo de cómo generar la imgen de CROCO la realiza Sepúlveda (2023) de la siguiente forma:

```
FROM ubuntu:16.04
# FROM debian:10-slim
LABEL maintainer="andres.sepulveda@gmail.com"
# Based on Dockerfiles from
# RSoutelino (http://github.com/metocean/docker-roms-public)
# Solene Le Gac (http://forum.croco-ocean.org/question/708/croco-i

RUN apt-get update \
&&  apt-get upgrade -y \
&&  apt-get install gfortran -y \
&&  apt-get install make -y \
&&  apt-get install libopenmpi-dev -y \
&&  apt-get install libhdf5-openmpi-dev -y \
&&  apt-get install libnetcdf-dev -y \
&&  apt-get install libnetcdff-dev -y \
&&  apt-get install bc -y \
&&  apt-get install git -y \
&&  apt-get install curl -y \
&&  apt-get install netcdf-bin -y \
&&  apt-get install nco -y \
&&  apt-get install unzip -y \
&&  apt-get install wget -y \
&&  apt-get install sudo -y \
&&  apt-get install vim -y \
&&  apt-get install octave -y \
&&  apt-get install octave-octcdf -y \
```

```
&&  apt-get clean -y
RUN useradd -m croco && echo "croco:croco" | chpasswd && adduser
RUN mkdir -p /home/croco
RUN chown -R croco:croco /home/croco
USER croco
WORKDIR /home/croco/
RUN wget https://data-croco.ifremer.fr/CODE_ARCHIVE/croco-v1.3.ta
RUN gzip -d croco-v1.3.tar.gz
RUN tar -xvf croco-v1.3.tar
RUN rm croco-v1.3.tar
RUN wget https://data-croco.ifremer.fr/CODE_ARCHIVE/croco_tools-v
RUN gzip -d croco_tools-v1.3.tar.gz
RUN tar -xvf croco_tools-v1.3.tar
RUN rm croco_tools-v1.3.tar
#
# All external datasets (8.9GB)
#
WORKDIR /home/croco/croco_tools
RUN wget https://data-croco.ifremer.fr/DATASETS/DATASETS_CROCOTOOL
RUN gzip -d DATASETS_CROCOTOOLS.tar.gz
RUN tar -xvf DATASETS_CROCOTOOLS.tar
RUN rm DATASETS_CROCOTOOLS.tar
RUN mv DATASETS_CROCOTOOLS/* .
RUN rm -rf DATASETS_CROCOTOOLS
WORKDIR /home/croco/
```

A4.2. Creación entorno en clúster Kubernetes

Para utilizar Kubernetes y los recursos de nuestro clúster de Kubernetes, podemos utilizar la herramienta kubectl que nos proporciona diferentes opciones para aplicar, obtener información, eliminar recursos, etc. De esta forma, se pondrá de ejemplo los comandos más útiles a la hora de querer configurar nuestro clúster o solicitar información de estado de nuestros pods, nodos y contenedores. Aquí no se expondrá como realizar un clúster Kubernetes, aunque estos comandos fueron utilizados a modo de prueba en un clúster Kubernetes de GCP.

Una vez ya montado nuestro servidor podemos crear un espacio de nombre para realizar nuestras aplicaciones, para esto se realiza el siguiente comando

```
kubectl create ns croco
```

create, no ayuda a crear recursos. *ns* es el name space y lo que sigue es el nombre de nuestro *ns*.
Una vez creado nuestro *ns*, será necesario generar nuestro volumen persistente en donde se almacenarán nuestros datos

```
kubectl create -f create_pv.yaml
```

Luego generamos un *service account* de Kubernetes que permite el uso de los recursos para Kubernetes dentro del clúster

```
kubectl create -f create_svc_croco.yaml
```

Especificamos el volumen persistente para nuestros contenedores creados en pods mediante StatefulSet:

```
kubectl create -f create_pvc.yaml
```

Generamos la réplica de nuestros pods que contienen los contenedores que a su vez tienen la imagen que nosotros deseamos replicar, en este caso puede realizar

```
kubectl create -f stateful_croco_ssh_kube.yaml
```

Una vez creado este archivo podemos obtener el estado de nuestro pod, con esto podemos ver si se creo(aunque tomará un par de minutos)

```
kubectl get pods -n croco -o wide
```

Aunque si ocurren problemas podemos ver el estado de lo que está sucediendo con el comando *describe*, lo aplicamos de la siguiente forma:

```
kubectl describe pods -n croco
```

Una vez verificado todo este en correcto funcionamiento, podemos ingresar a alguno de nuestros pods podemos ingresar a nuestro contenedor

```
kubectl exec -it croco-ssh-statefulset02-0 --container croco-ssh
```

Si es solo una prueba para terminar la prueba, podemos eliminar nuestros pods o recursos con el comando *delete* de la siguiente manera:

```
kubectl delete statefulset croco-ssh-statefulset02 -n croco
```

A4.3. Archivos YAML

Estos archivos nos sirven para la aprisionar o liberar nuestros recursos en el clúster de Kubernetes, se expondrán algunos ejemplos que requieren de adaptación según el caso basados en Jung et al. (2020), algunos de estos son solicitados y fueron utilizados en la sección anterior:

- **Persistent Volume**: create_pv.yaml

```
    apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-prueba
  namespace: croco
  labels:
   name: pv-prueba
spec:
  capacity:
    storage: 20Gi
  accessModes:
    - ReadWriteMany
  storageClassName: standard
```

- **Persistent Volume Claim**; create_pvc.yaml

```
    apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-prueba
  namespace: croco
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
```

```
    storageClassName: standard
    volumeName: pv-prueba
```

- **StatefulSet**

```
    apiVersion: apps/v1
  kind: StatefulSet
  metadata:
    name: croco-ssh-statefulset02
    namespace: croco
    labels:
      app: croco-ssh
  spec:
    replicas: 1
    selector:
      matchLabels:
        app: croco-ssh
    serviceName: "croco-ssh"
    template:
      metadata:
        labels:
          app: croco-ssh
      spec:
        containers:
        - name: croco-ssh
          image: andressepulveda/croco_oceanv1.2.1b
          resources:
           limits:
            cpu: 940m
           requests:
            cpu: 228m
          volumeMounts:
          - mountPath: mydisk
            name: pv-prueba
          command: ["/bin/sh", "-c"]
          args:
```

```
        - echo starting;
          /usr/sbin/sshd;
          sleep 360000;
          echo done;
      ports:
      - containerPort: 22
    volumes:
    - name: pv-prueba
      persistentVolumeClaim:
        claimName: pvc-prueba
```

- **ServiceAccount**: create_svc_croco.yaml

```
    apiVersion: v1
kind: ServiceAccount
metadata:
 name: sa-croco
 namespace: croco
```

A5. Pruebas Google Cloud Plataform

En trabajos previos se realizaron pruebas de rendimiento de CROCO para MV en GCP, para esto se editan archivos del código fuente de CROCO para no grabar salidas avg y his. Esto se realiza en el sentido para evaluar realmente el rendimiento de la MV y no verse influenciado por el grabado de las salidas.
A diferencia de Azure, GCP ofrece 300 USD de prueba con un límite de 8 vCPU, por lo que permite realizar más pruebas de distribución. Se usa una máquina de 8vCPU de la serie E para procesos estándar, con 16 GB de RAM y un disco de almacenamiento HDD de 10 GB. Con un coste de 0.30 USD/hora.

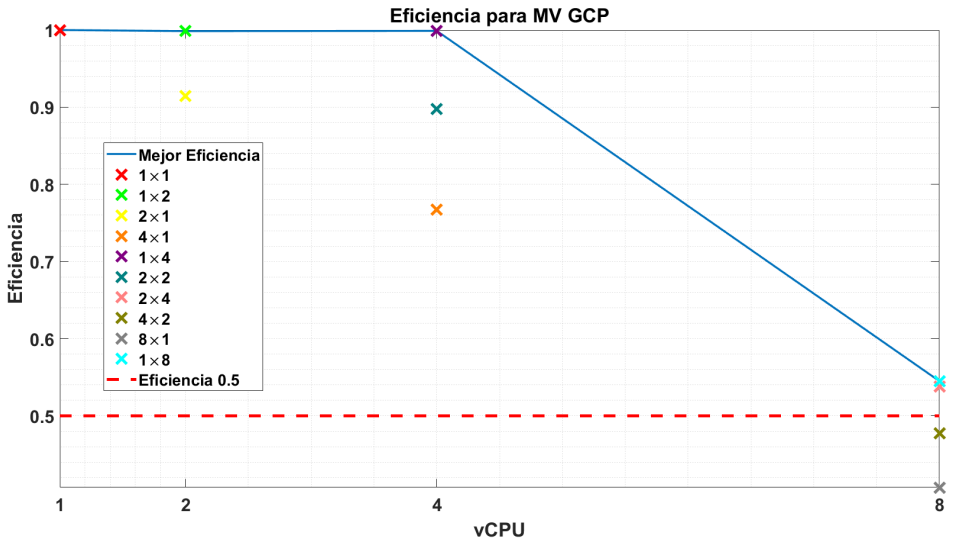


Figura A5.1: Eficiencia para simulación de 1 día sin guardado de salida en MV e2-standard-8 de GCP

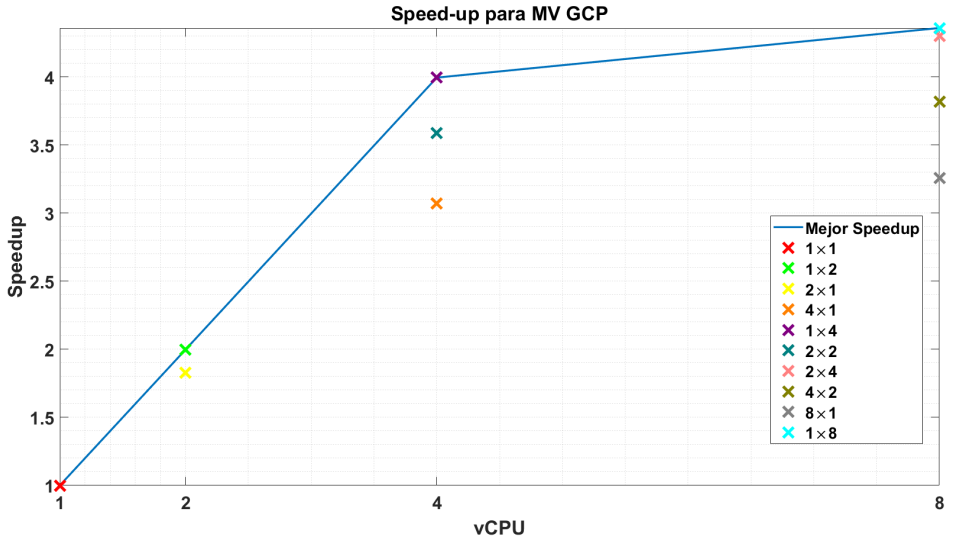


Figura A5.2: Speed-up para simulación de 1 día sin guardado de salida en MV e2-standard-8 de GCP

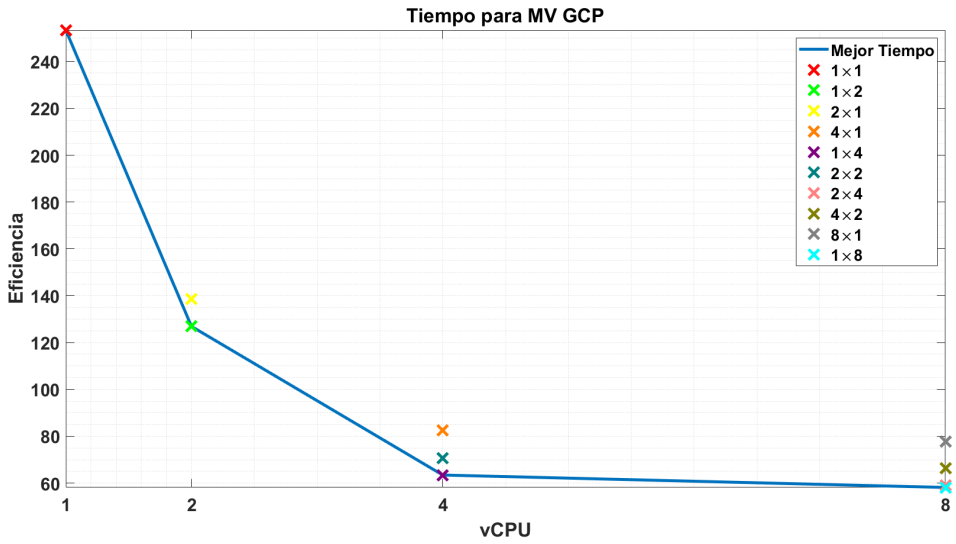


Figura A5.3: Tiempo de calculo para simulación de 1 día sin guardado de salida en MV e2-standard-8 de GCP

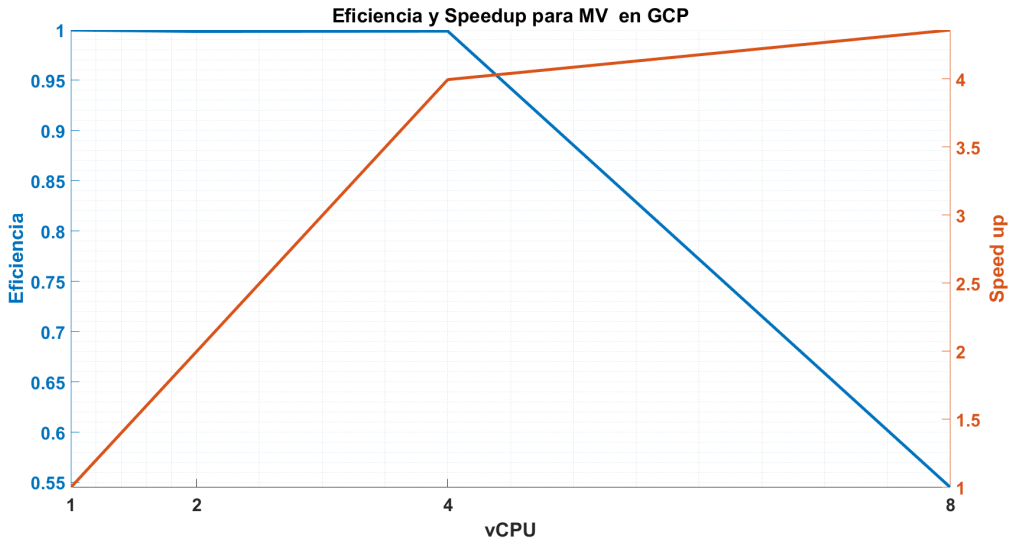


Figura A5.4: *Eficiencia* y *Speedup* simulación de 1 día sin guardado de salida en MV e2-standard-8 de GCP

A6. Pruebas D4as_v4

Se realiza la misma prueba con motivo de comparación utilizando el mismo disco SSD de la MV de F4s_v2, 4 vCPU y 16 GiB de RAM. Este tiene un costo de 0.875 USD/hora.

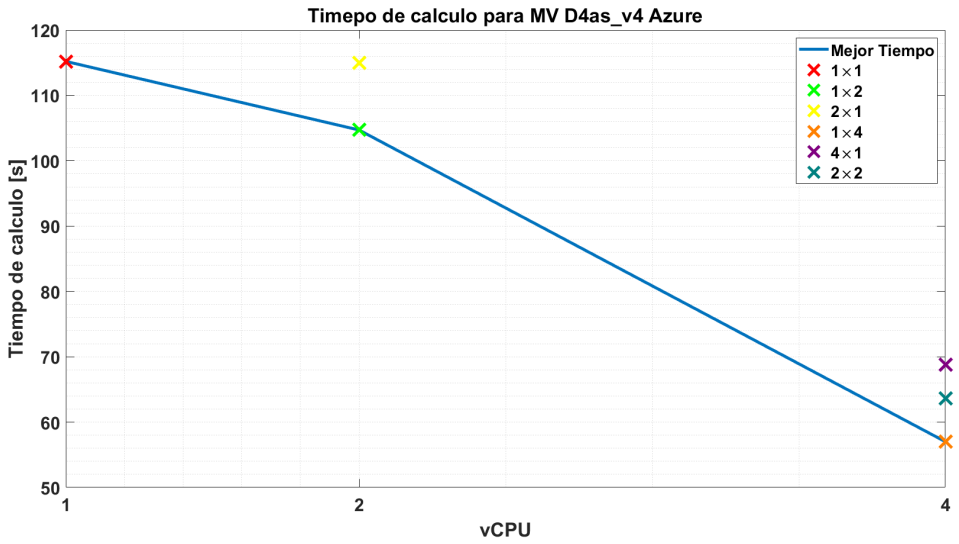


Figura A6.1: Tiempo de calculo para simulación de 1 día en MV D4as_v4 de Azure.

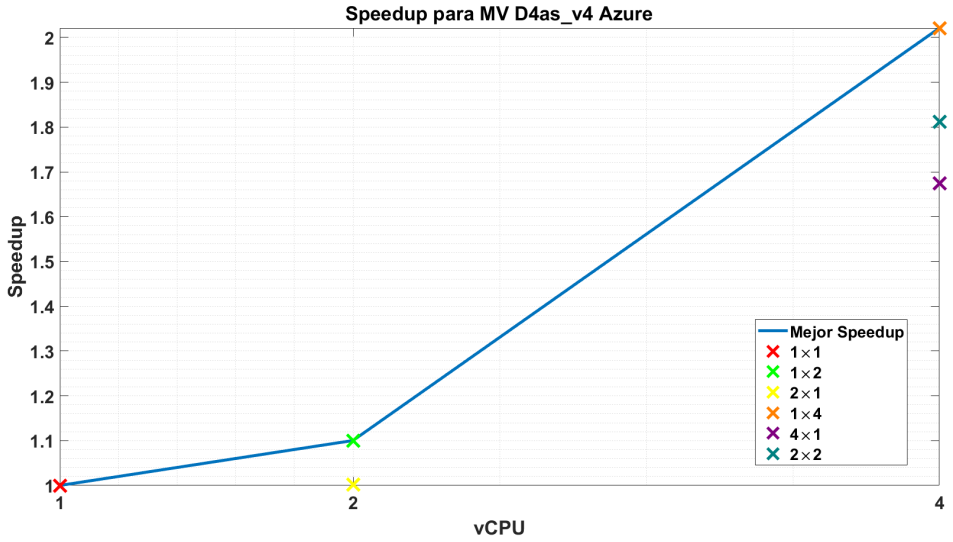


Figura A6.2: *Speed-up* para simulación de 1 día en MV D4as_v4 de Azure.

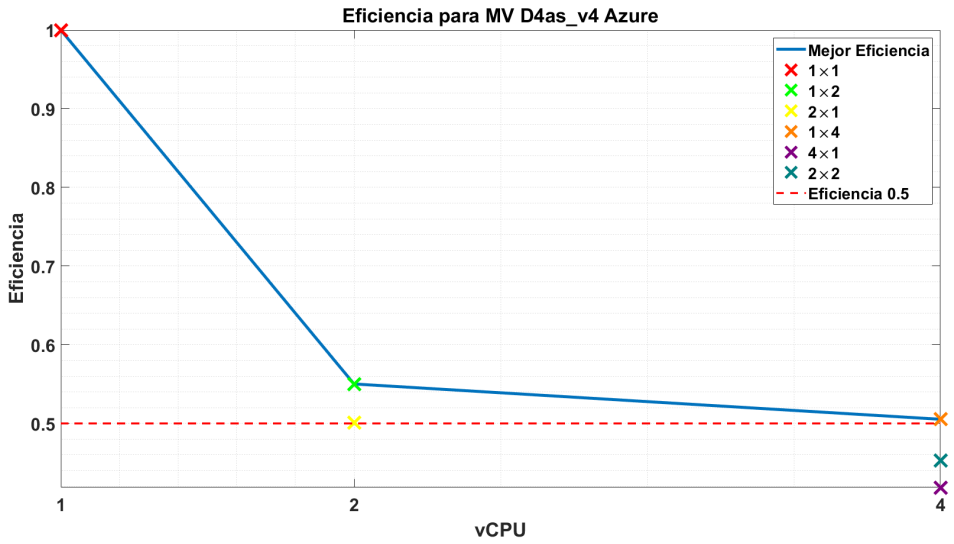


Figura A6.3: *Eficiencia* y *Speedup* para simulación de 1 día en MV D4as_v4 de Azure.

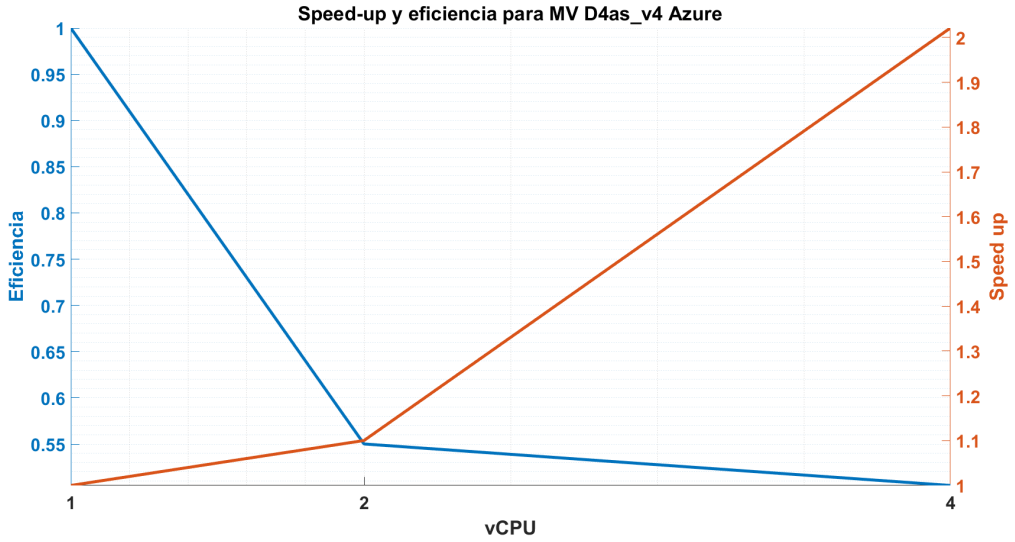


Figura A6.4: *Eficiencia* para simulación de 1 día en MV D4as_v4 de Azure.

A6.1. Comparación

Se realiza una comparación de los valores de tiempo de cómputo, *speed-up* y eficiencia entre la serie F4s_v2 y D4as_v4.

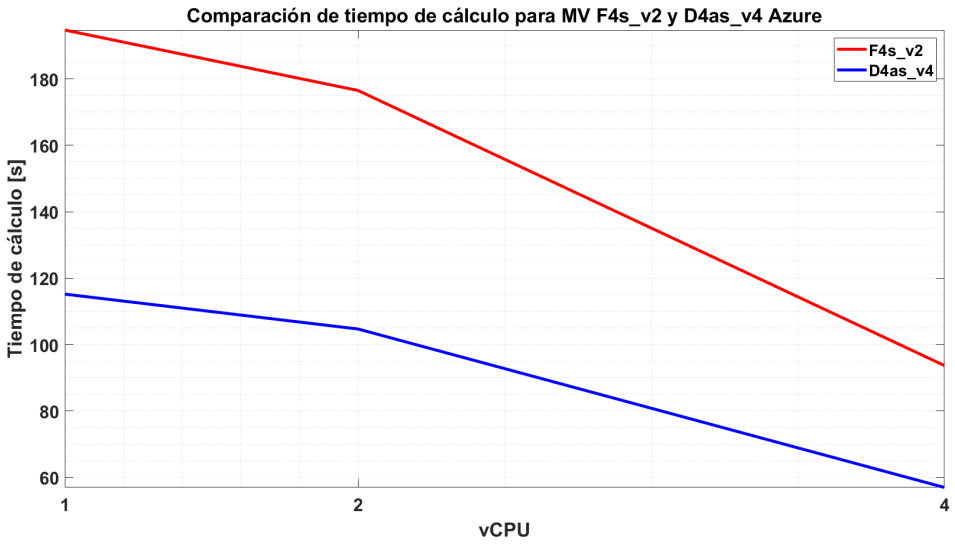


Figura A6.5: Tiempo de cómputo entre la serie F4s_v2 y D4as_v4 para 1 día de simulación.

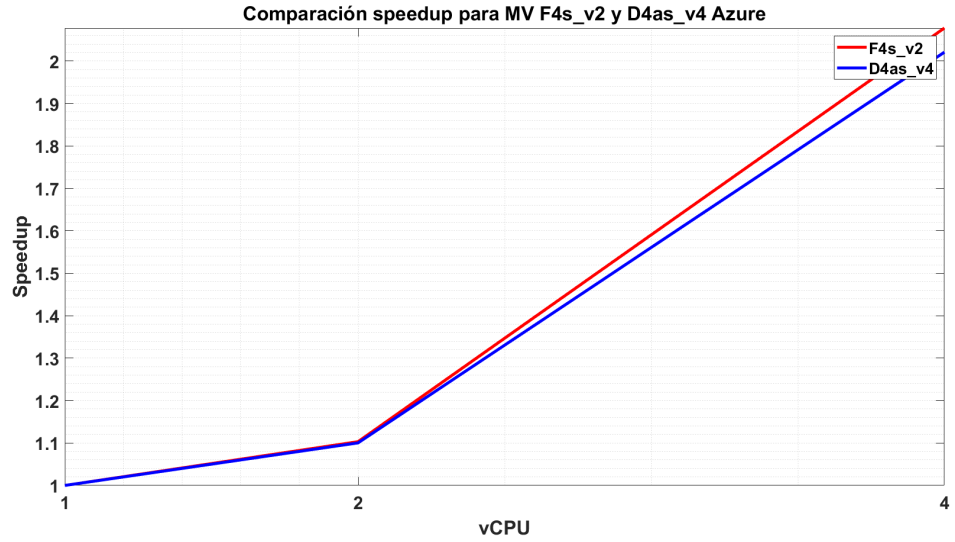


Figura A6.6: *Speed-up* entre la serie F4s_v2 y D4as_v4 para 1 día de simulación.

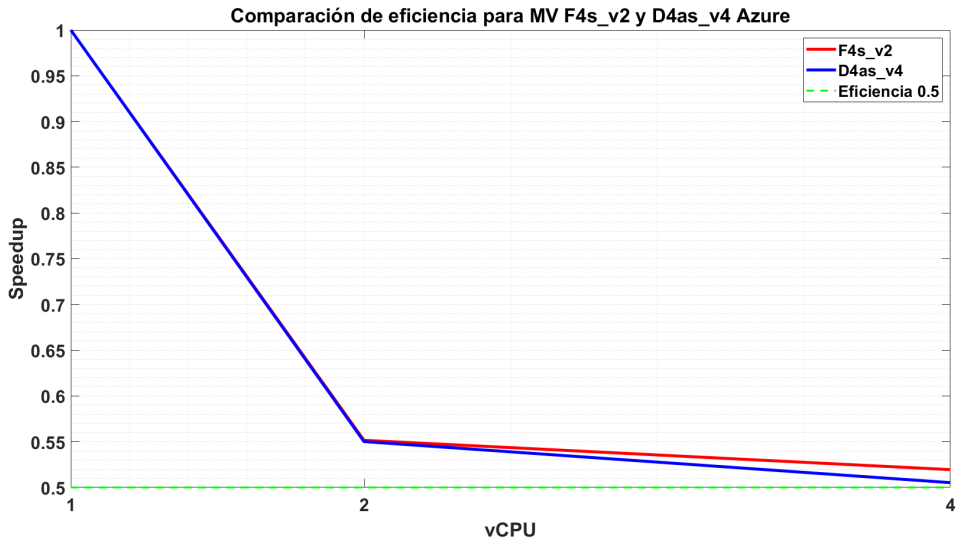


Figura A6.7: *Eficiencia* entre la serie F4s_v2 y D4as_v4 para 1 día de simulación.

A7. Azure MPI

Lockwood (2022), explica una forma de generar un clúster para ejecutar trabajos MPI de la siguiente manera. Primeramente, se puede crear un archivo para configurar paquetes necesarios ejecutar nuestro clúster (se pueden agregar más de los que se ejemplifica), este lo denominaremos cloud-init.txt, este contendrá lo siguiente:

```
#cloud-config
package_upgrade: true
packages:
  - clustershell
  - openmpi-bin
  - libopenmpi-dev
```

Creamos un grupo de recursos

```
az group create --name glocktestrg --location eastus
```

Creamos un grupo de colocación de proximidad(*Proximity placement group* o ppg). Esto ayuda a generar una red de burbuja de baja latencia en el caso si queremos crear más de una MV o clúster. Se realiza de la siguiente forma:

```
az ppg create --name glocktestppg \
              --resource-group glocktestrg \
              --intent-vm-sizes Standard_DS1_v2
```

Luego solicitamos nuestras MVs de la siguiente forma.

```
az vm create --name clusterej \
             --resource-group glocktestrg \
             --image UbuntuLTS \
             --ppg glocktestppg \
             --generate-ssh-keys \
             --size Standard_DS1_v2 \
             --accelerated-networking true \
             --custom-data cloud-init.txt \
             --count 2
```

Donde cada sección no indicaría lo siguiente: Esto quiere decir:

- –**name** Se designarán nombres numerados como ejemplo clusterej0, clusterej1, etc.
- –**image UbuntuLTS:** instala Ubuntu en cada máquina virtual.
- –**generate-ssh-keys:** Coloca su clave SSH local en el archivo en todos los nodos que está creando /.ssh/id_rsa.pubauthorized_keys
- –**size Standard_DS1_v2:** el tipo de MV DS1_v2
- –**accelerated-networking true:** proporciona una latencia de red ultrabaja constante a través del hardware programable interno de Azure y tecnologías como SR-IOV, no tiene un costo adicional.
- –**custom-data cloud-init.txt:** generar datos personalizados que se generarán al inicio de nuestra MVs.
- –**count 4:** número de MVs totales

Una vez creado todo, podemos visualizar todo lo creado(recursos) con el siguiente comando:

```
az resource list --resource-group glocktestrg -o table
```

Arrojando como resultado

Name	ResourceGroup	Location	Type	Status
glocluster0_OsDisk_1_2bb165748cb34ebbac53a2a1b22605ac	GLOCKTESTRG	eastus	Microsoft.Compute/disks	
glocluster1_OsDisk_1_0964837624294b29abf4bffe735bb8b8	GLOCKTESTRG	eastus	Microsoft.Compute/disks	
glocktestppg	glocktestrg	eastus	Microsoft.Compute/proximityPlacementGroups	
glocluster0	glocktestrg	eastus	Microsoft.Compute/virtualMachines	
glocluster1	glocktestrg	eastus	Microsoft.Compute/virtualMachines	
glocclusterVMNic0	glocktestrg	eastus	Microsoft.Network/networkInterfaces	
glocclusterVMNic1	glocktestrg	eastus	Microsoft.Network/networkInterfaces	
glocclusterNSG	glocktestrg	eastus	Microsoft.Network/networkSecurityGroups	
glocclusterPublicIP0	glocktestrg	eastus	Microsoft.Network/publicIPAddresses	
glocclusterPublicIP1	glocktestrg	eastus	Microsoft.Network/publicIPAddresses	
glocclusterVNET	glocktestrg	eastus	Microsoft.Network/virtualNetworks	

Figura A7.1: Resultado en consola para la visualización de los recursos.

Solo nos queda configurar las Claves SSH respectivas entre máquinas.

A7.1. Cotización

Para todas las empresas que ofrecen servicios en la nube, se realiza una cotización para estimar un aproximado de la utilización de solo una máquina virtual optimizada para procesos, este tipo de instancia esta enfocada en la optimización y eficiencia para aprovechar el máximo rendimiento al recurso. Esta prueba se hace para estimar un valor de 1 mes de uso continuo(720 horas) con un almacenamiento de SSD estándar de 120 GB. Para esto se cotiza en la calculadora de precios de Azure, AWS y GoogleCloud:

Nubes	MV	Procesadores	Almacena-miento	Precio Estimado
Azure	Series Fv2: 32 vCPU, 64GB de RAM	Procesadores de 3.ª generación Intel® Xeon® Platinum 8370C (Ice Lake), Intel® Xeon® Platinum 8272CL (Cascade Lake) o Intel® Xeon® Platinum 8168 (Skylake). Presenta una velocidad de reloj turbo sostenida de todos los núcleos de hasta 3,4 GHz y una velocidad de reloj turbo de un solo núcleo máxima de 3,7 GHz	128 GB SDD Standar	1247.68 USD/Mes
AWS	Series c5: 36 vCPU, 72 GiB de RAM	Procesadores Intel Xeon personalizados de segunda generación con escala ajustable (Cascade Lake) con una frecuencia Turbo estable en todos los núcleos de 3,6 GHz y una frecuencia Turbo de núcleo individual máxima de 3,9 GHz.	120 GB SSD	1126.50 USD/Mes
Google Cloud	Series C2: 30vCPU, 96GB de RAM	Procesadores escalables Intel® Xeon® de 2ª generación que ofrece frecuencia turbo de nucleo máximo continuio de hasta 3.9GHz	124 GB SSD PD	1292.97 USD/Mes

Cuadro A7.1: Cotización realizada para una máquina virtual con 1 mes aproximado de uso continuo tres servicios de Nube estimado por el pago por uso(realizado el 26 de Marzo del 2023).

Los precios pueden actualizarse, además hay que tener en consideración los cambios de moneda que puede provocar una variación en el costo final. Para la realización de esta cotización se tuvo en cuenta como referencia las series Fsv2 de Azure, que están optimizadas para procesamiento y un mejor rendimiento en las cargas de trabajo de procesamiento vectorial en las operaciones de número de punto flotante de precisión individual y doble. Siendo precisas y óptimas para cargas de trabajos de cálculo.

Para las máquinas virtuales de los otros servicios de la nube varían en cuanto al escalamiento de las MV y su cantidad de vCPU, ambas series son también optimizadas para procesamiento. Para realizar una comparación aproximada se escogen procesadores similares teniendo valores aproximados en cuanto a su frecuencia mayor de los núcleos, pero no se encuentra en AWS la frecuencia más estable.

La serie C2 de GoogleCloud cuentan entre 4 y 60 vCPU, ofrece hasta 240GB de memoria RAM y 3 TB de almacenamiento local. Tienen un ancho de banda de red más alto de 50Gbps y 100Gbps. Dentro de las cargas de trabajo ideales para computación de alto rendimiento(HPC), optimizan las cargas de trabajo para el rendimiento de un solo subproceso, en especial con respecto al punto flotante.

En cuanto a la serie de MV de AWS, se tuvo en consideración un procesador similar(procesador Intel), pero dentro de las optimizadas para procesamiento existe una mayor gama con procesadores creados por AWS(procesadores Gravitón), por lo se selecciona la linea C5 que también esta considerada para la modelación científica y el computo de alto rendimiento, pudiendo escalar desde 2vCPU a 96 vCPU y desde 4 GiB hasta 192 GiB de memoria RAM, también para el ancho de banda varía pero podemos optar hasta 25 Gbps.

Podemos notar que en cuanto a los precios por mes, se encuentran en valores bastante similares siendo la más económica AWS y GC la de mayor precio, aunque tampoco podemos hacer una comparación tan exacta, ya que como vemos varía en la cantidad de procesadores(también en el tipo de procesador designado), la memoria RAM y también en el almacenamiento. Esto sin considerar el soporte ni los datos de salida que podemos generar.

A8. Tablas tiempo de cálculo, *speedup* y eficiencia

A8.1. MV F4s_v2 Azure

Dominios	Tiempo[s]	Eficiencia	Speedup
1×1	194.7	1	1
1×2	176.5	0.55	1.09
2×1	180.5	0.54	1.07
1×4	93.7	0.52	2.07
4×1	106	0.46	1.83
2×2	95.8	0.5	2.03

Cuadro A8.1: Valores del tiempo de cómputo, eficiencia, *speedup* para los dominios de la simulación de 1 día realizada en la MV F4s_v2 de Azure.

A8.2. MV Das_v4 Azure

Dominios	Tiempo[s]	Eficiencia	Speedup
1×1	115.2	1	1
1×2	104.7	0.55	1.1
2×1	115	0.5	1
1×4	57	0.50	2.02
4×1	68.8	0.42	1.67
2×2	63.6	0.45	1.81

Cuadro A8.2: Valores del tiempo de cómputo, eficiencia, *speedup* para los dominios de la simulación de 1 día realizada en la MV Das_v4 de Azure.

A8.3. MV e2-standard-8 GCP

Dominios	Tiempo[s]	Eficiencia	Speedup
1×1	253.44	1	1
1×2	126.9	0.99	1.99
2×1	138.59	0.91	1.82
4×1	82.58	0.76	3.07
1×4	63.43	0.99	3.99
2×2	70.61	0.89	3.59
2×4	58.92	0.53	4.30
4×2	66.39	0.47	3.81
8×1	77.8	0.40	3.25
1×8	58.13	0.54	4.36

Cuadro A8.3: Valores del tiempo de cómputo, eficiencia, *speedup* para los dominios de la simulación de 1 día realizada en la MV e2-standard-8 de GCP.