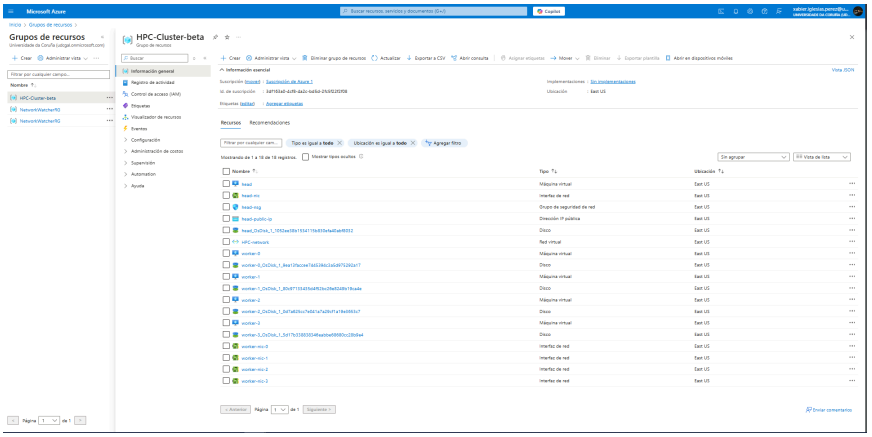
Práctica opcional: Despliegue de un clúster con herramientas IaC y ejecución de NPB en diferentes proveedores cloud

Xabier Iglesias Pérez (xabier.iglesias.perez@udc.es)
Version 1.0 - GIT: https://github.com/TretornESP/HPCN-TERRAFORM

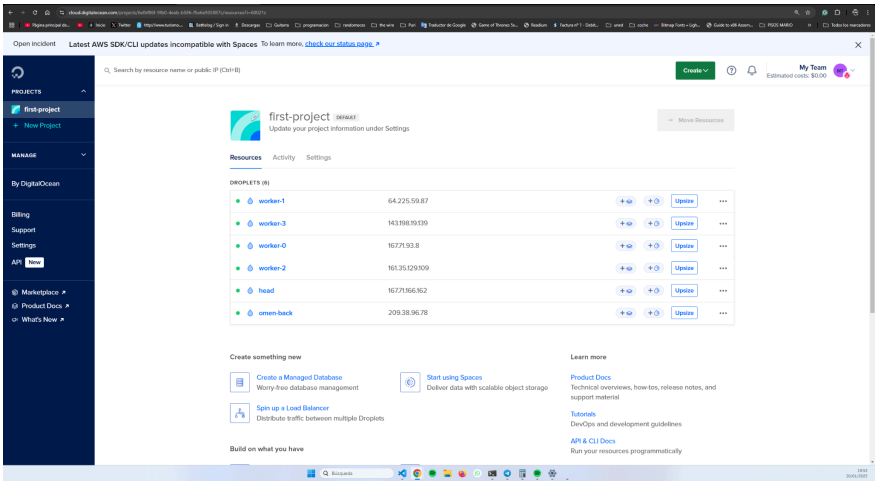
1. Introducción

Este proyecto es la continuación natural de la práctica de esta asignatura sobre clústeres virtuales en AWS. Siendo similar en su planteamiento, la principal diferencia es que el objetivo de aprendizaje es familiarizarse con otras plataformas de computación en la nube así cómo con tecnologías de *infraestructura como código (IaC)*. Para ello hemos creado un nuevo módulo de automatismos que enlaza con el fichero *execute.sh* de la práctica anterior. y que, mediante Terraform, es capaz de desplegar un clúster semejante en diferentes nubes. A la fecha de redacción de este borrador hemos ejecutado pruebas exitosas en Azure, GCP y DigitalOcean.

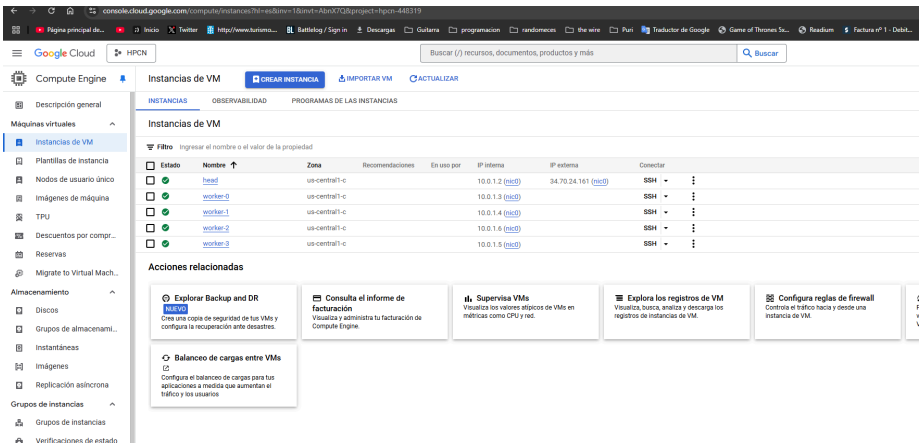
Cómo evidencia adjuntamos unas breves capturas:



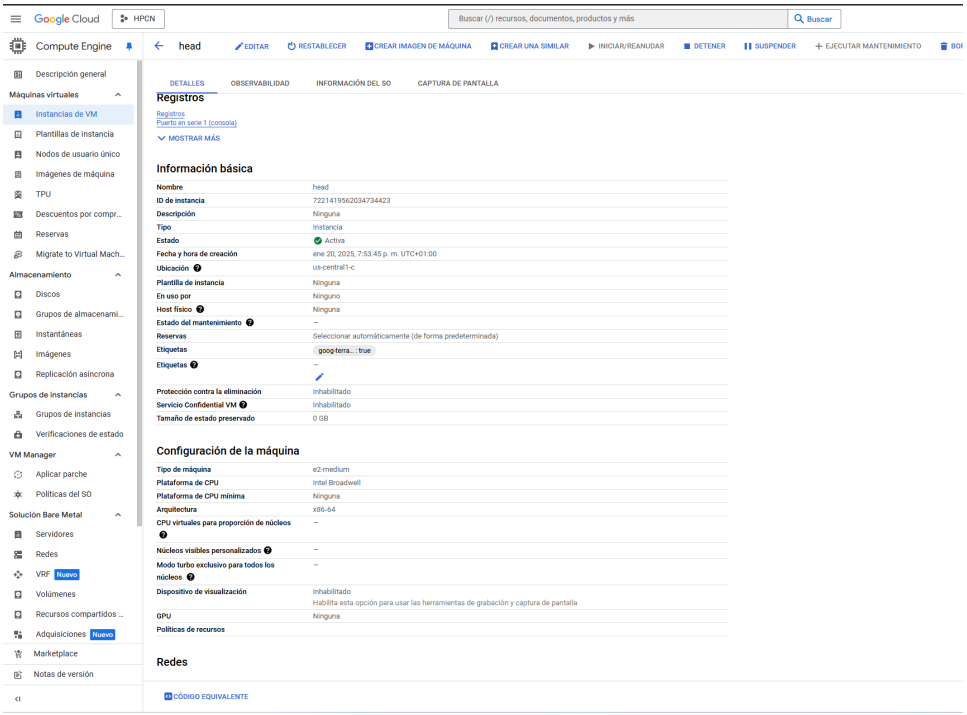
Captura 1. Pantalla de visor de recursos agregada de Azure.



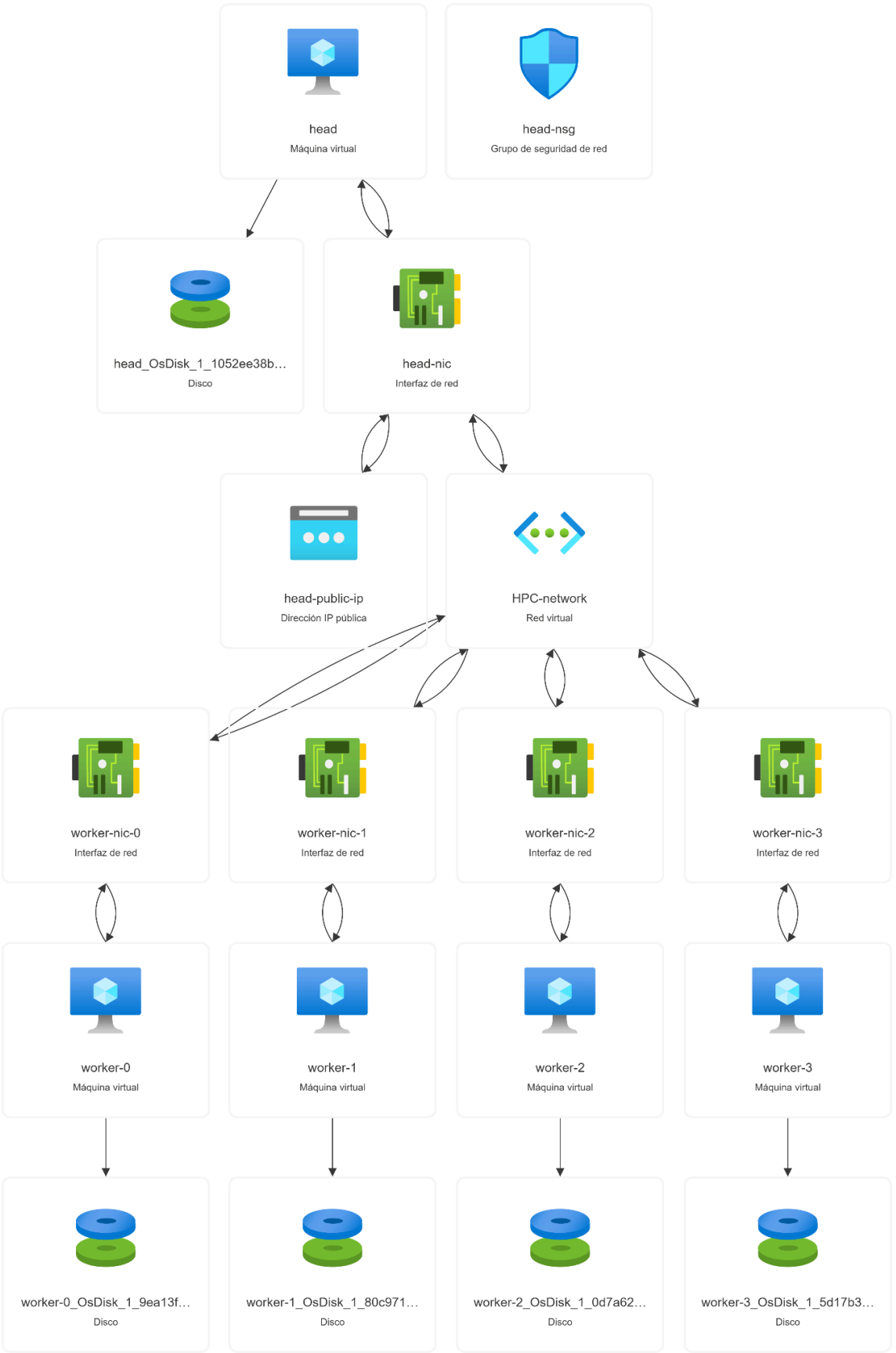
Captura 2. Instancias (Droplets) en DigitalOcean



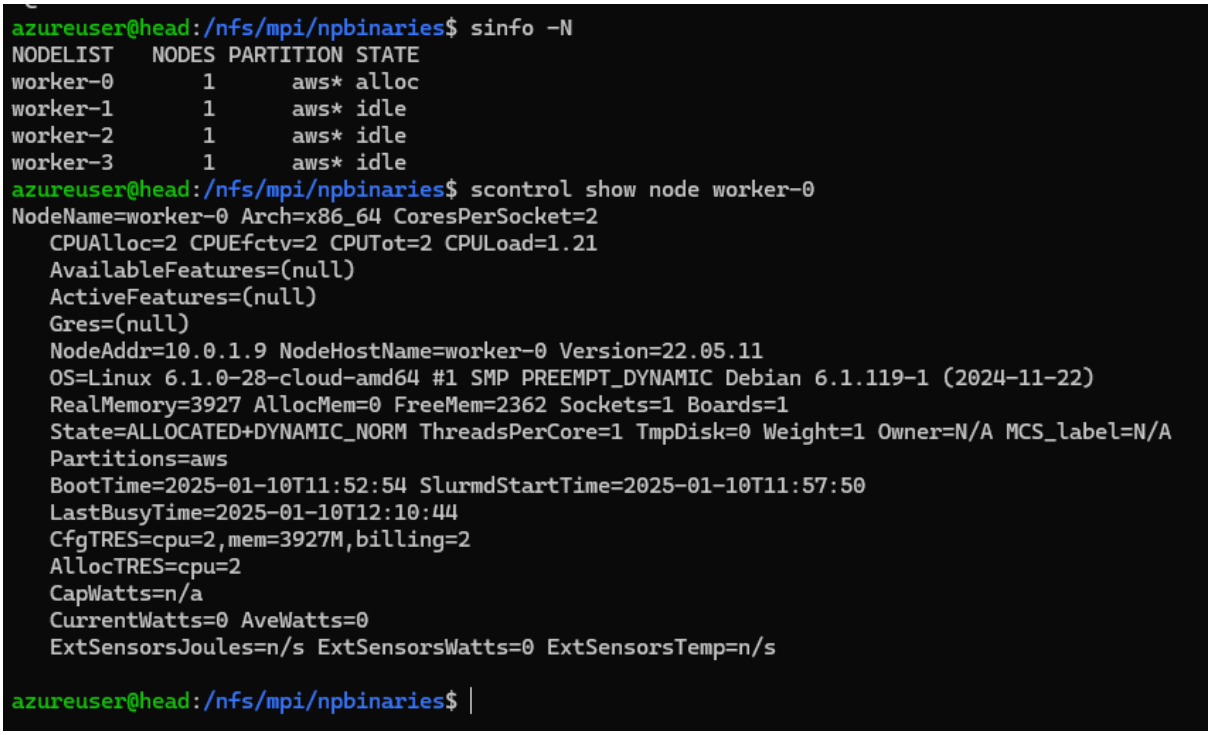
Captura 3. Instancias en GCP



Captura 4. Nodo head en GCP



Captura 5. Arquitectura del clúster generado con terraform (Según Azure, pero equivalente en todas los proveedores)



Captura 6. Output de los comandos sinfo y scontrol sobre el head node y un worker

2. Configuración experimental

En este caso, si bien nos guiamos por un criterio de similitud con respecto a la práctica, estamos limitados a un máximo de 10 vcpus, con lo que nuestra configuración será de 4 nodos con 2vcpus para los workers y 4gb de RAM. Mientras que el head node consumirá las 2vcpus y 4 gb de ram restantes. En sus diferentes providers emplearemos los siguientes tipos de instancia.

Provider	Instancia	Características
GCP	e2-medium	2vCores x 4GB de RAM
Azure	B2s	2vCores x 4GB de RAM
DO	s-2vcpu-4gb	2vCores x 4GB de RAM

Atendiendo a criterios de coste emplearemos instancias generalistas (aún así hemos tenido que pagar unos 80 euros en cómputo porque nos quedamos dormidos con las instancias Azure encendidas). Para estas pruebas todas las instancias de cada provider se encontrarán en la misma zona de disponibilidad (ó conceptos análogos). Todos los resultados son la media de 5 ejecuciones.

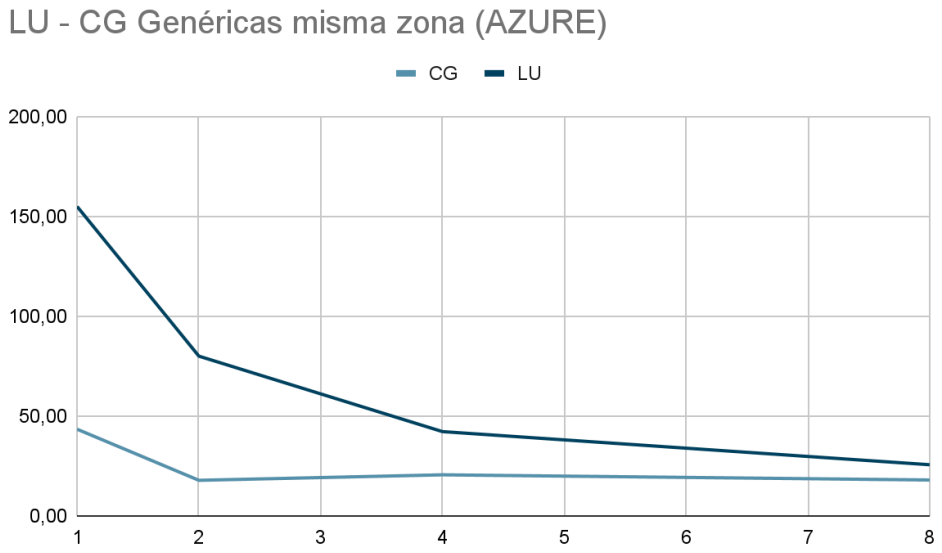
En lo que al sistema operativo se refiere, emplearemos Debian 12 vanilla sobre el que desplegaremos SLURM 22.05 y OpenMPI 4.1.4. Al igual que en la práctica emplearemos el script de automatismos execute.sh, a través de la interfaz runner.sh

3. Experimentos

De cara a centrar el tiro en la comparación entre proveedores, hemos reducido el experimento a ejecuciones de las cargas de trabajo CG - Conjugate Gradient y LU - Lower-Upper Gauss-Seidel Solver sobre instancias de cómputo genéricas y sobre la misma zona de disponibilidad.

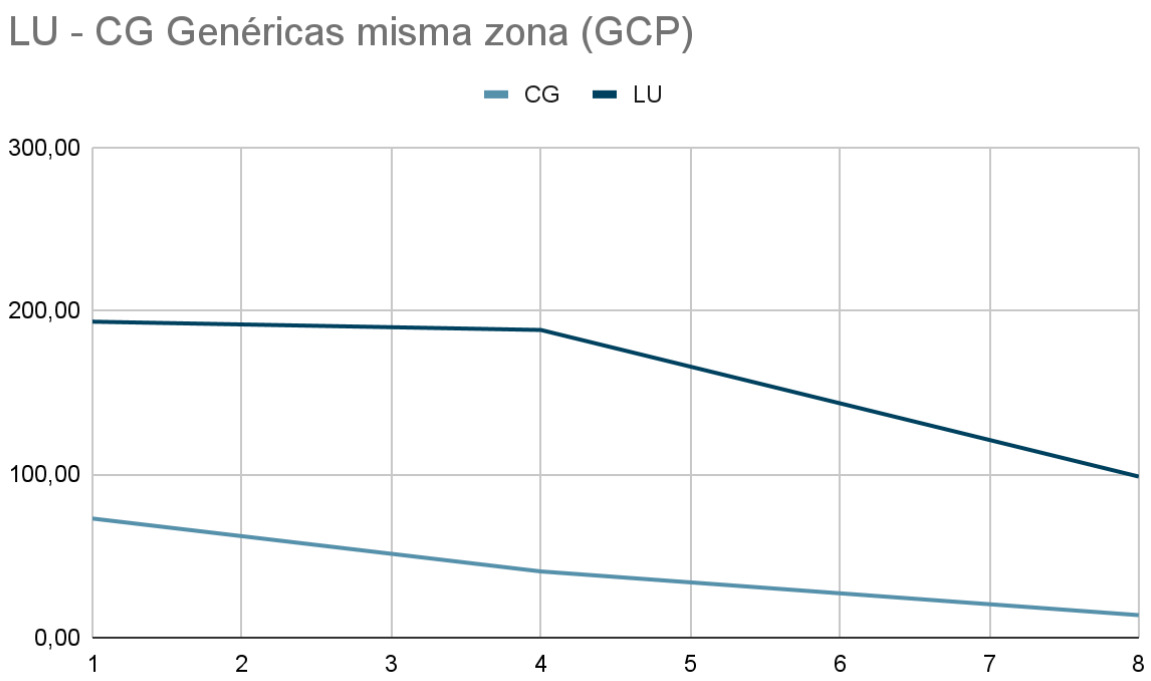
Hemos avanzado desde la versión secuencial 1 nodo / 1 core hasta la configuración con el máximo paralelismo que nos permiten los límites de cuota actuales 4 nodos / 8 cores) y hemos obtenido los siguientes resultados.

3.1. Azure $0.042178 usd/hour



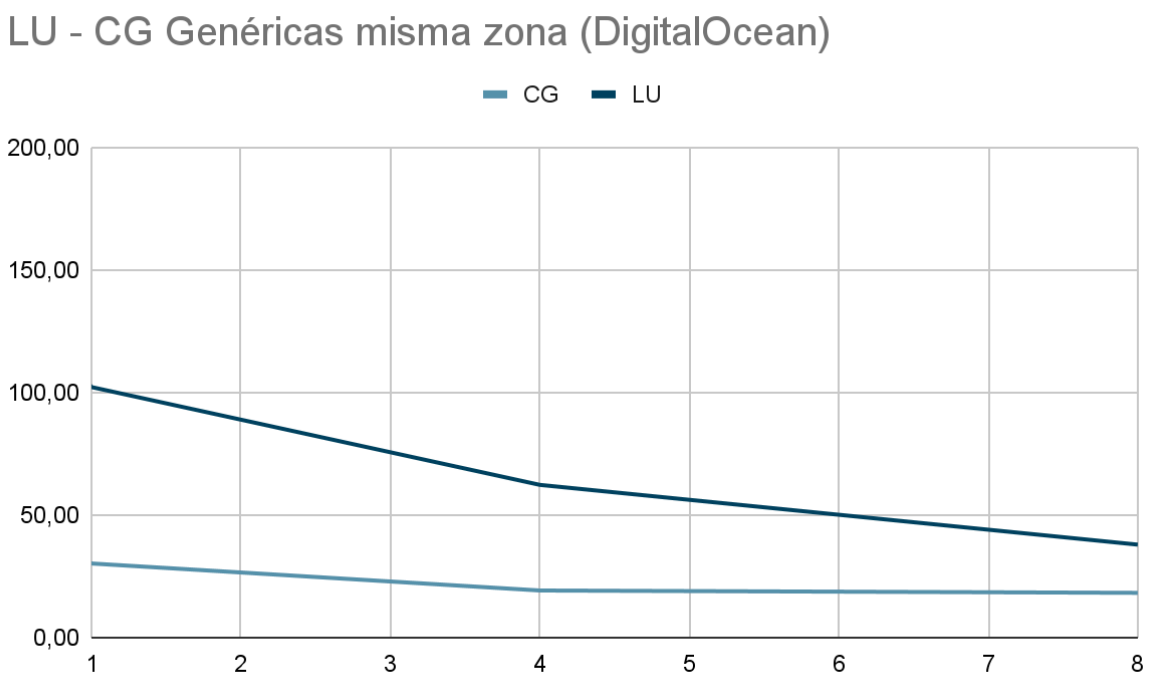
Inst. Generales	1 nodo (1 core, control)	1 nodo (2 cores)	2 nodos (4 cores)	4 nodos (8 cores)
CG				
Tiempos	43,416	24,676	20,580	17,988
Speedup	1	1,759	2,110	2,414
Ef. Paralela	1	0,880	0,527	0,302
LU				
Tiempos	154,916	79,990	42,214	25,630
Speedup	1	1,937	3,670	6,044
Ef.Paralela	1	0,968	0,917	0,756

3.2. GCP - $0.037731 usd/hour



Inst. Generales	1 nodo (1 core, control)	1 nodo (2 cores)	2 nodos (4 cores)	4 nodos (8 cores)
CG				
Tiempos	85,800	72,830	40,466	13,734
Speedup	1	1,178	2,120	6,247
Ef. Paralela	1	0,589	0,530	0,781
LU				
Tiempos	252,590	193,288	188,210	98,516
Speedup	1	1,307	1,342	2,564
Ef.Paralela	1	0,653	0,336	0,320

3.3. DigitalOcean $0.04762 usd/hour

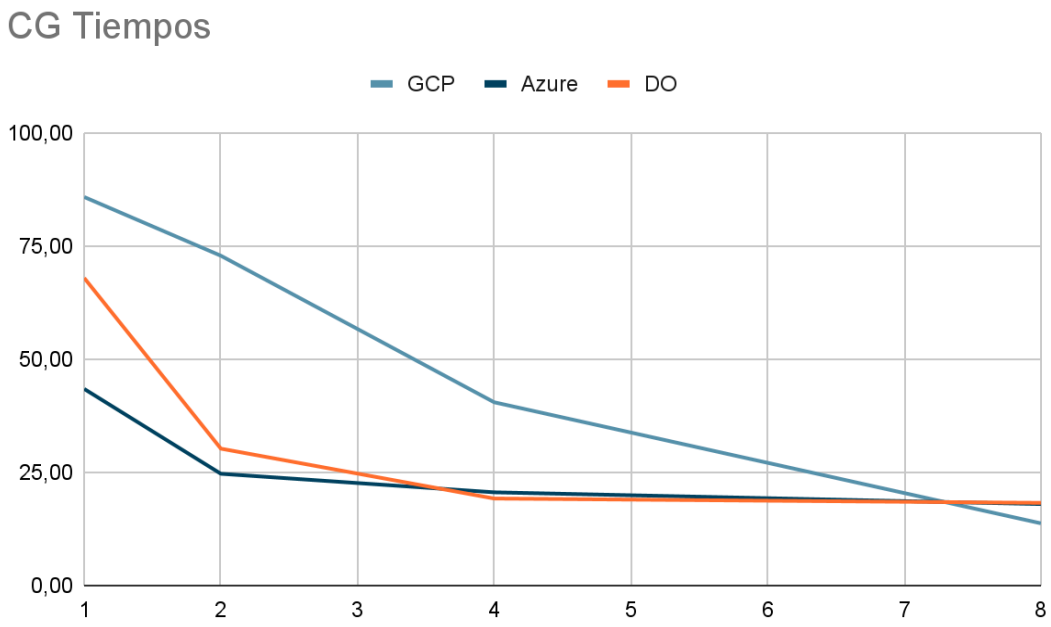


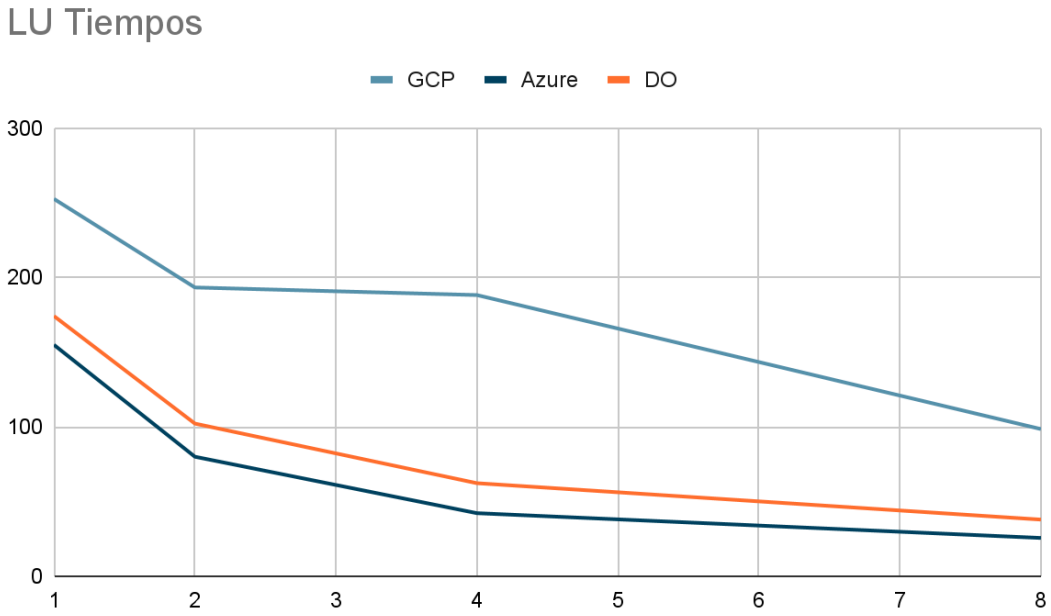
Inst. Generales	1 nodo (1 core, control)	1 nodo (2 cores)	2 nodos (4 cores)	4 nodos (8 cores)
CG				
Tiempos	67,922	30,224	19,208	18,252
Speedup	1	2,247	3,536	3,721
Ef. Paralela	1	1,124	0,884	0,465
LU				
Tiempos	174,184	102,176	62,286	37,918
Speedup	1	1,705	2,797	4,594
Ef.Paralela	1	0,852	0,699	0,574

4. Conclusiones

En base a los datos en bruto del apartado 3 hemos realizado un análisis comparativo en base a eficiencia bruta y en precio / potencia. Así observamos que:

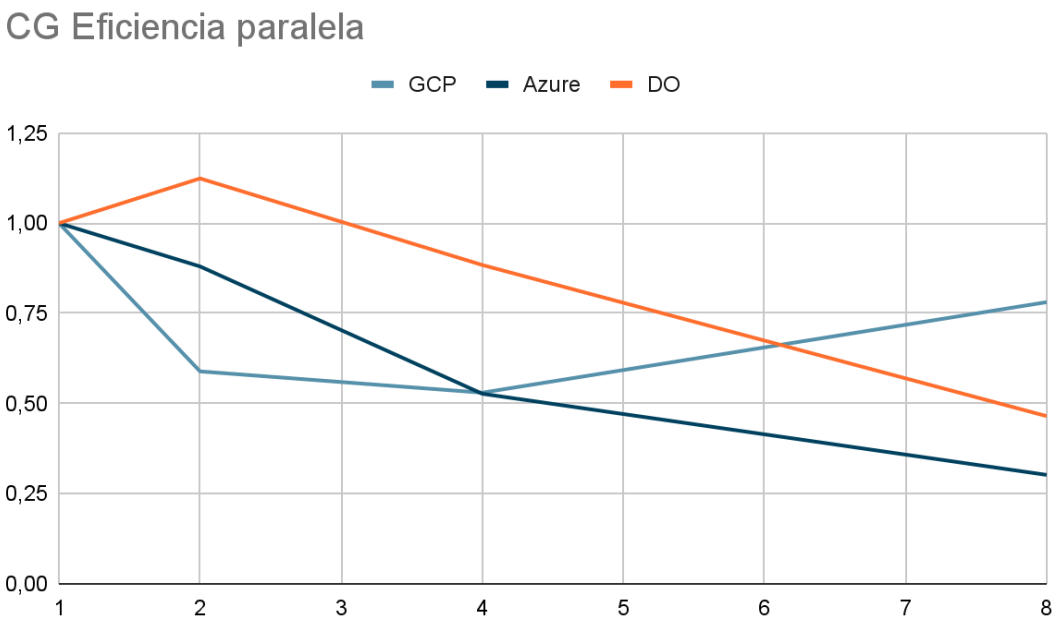
4.1. Comparativa de tiempos

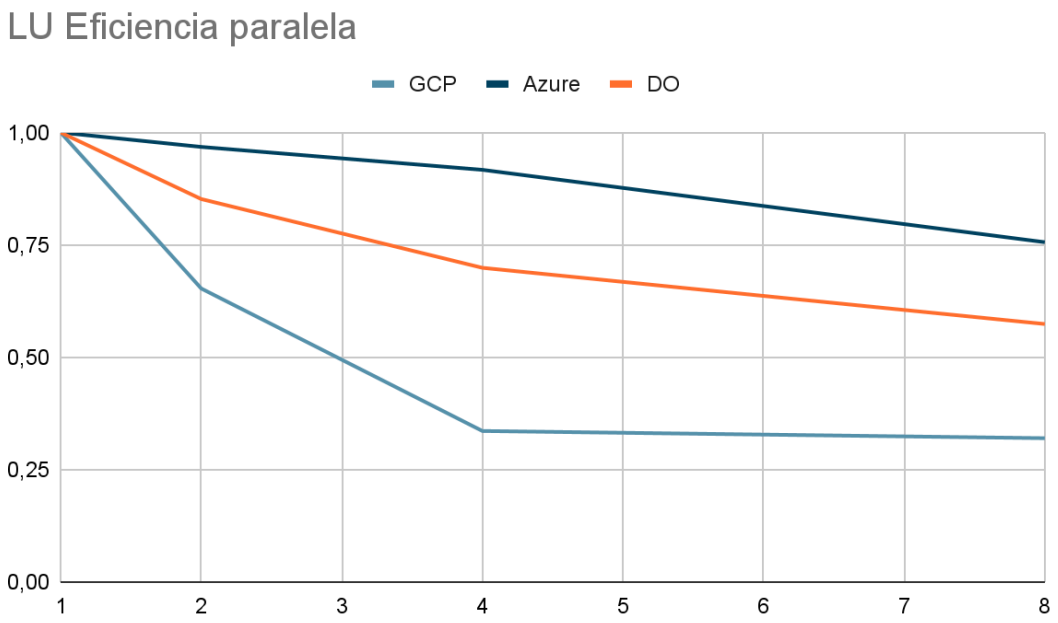




En ambos casos el claro ganador en escalabilidad es Google, con un descenso casi lineal, mientras que Azure y DO se comportan de forma casi idéntica, con un cuello de botella muy importante cuando empezamos a trabajar con nodos remotos. Sin embargo la potencia bruta de las CPUs de Azure y DO es también claramente superior a las de Google.

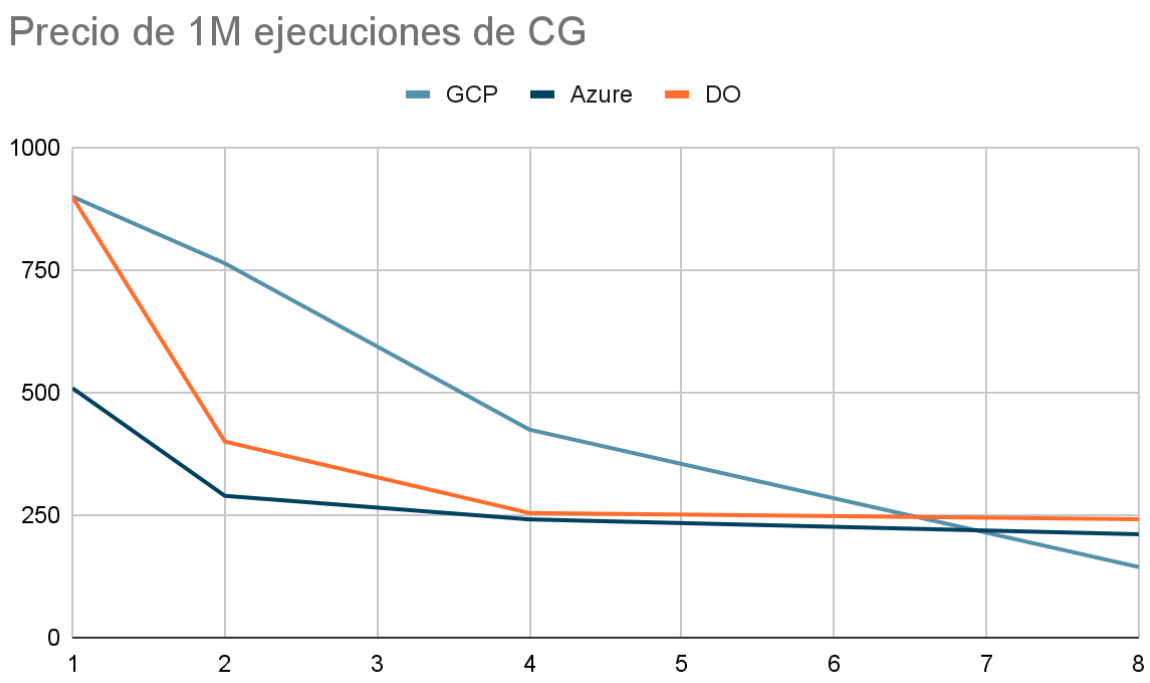
4.2. Eficiencia paralela

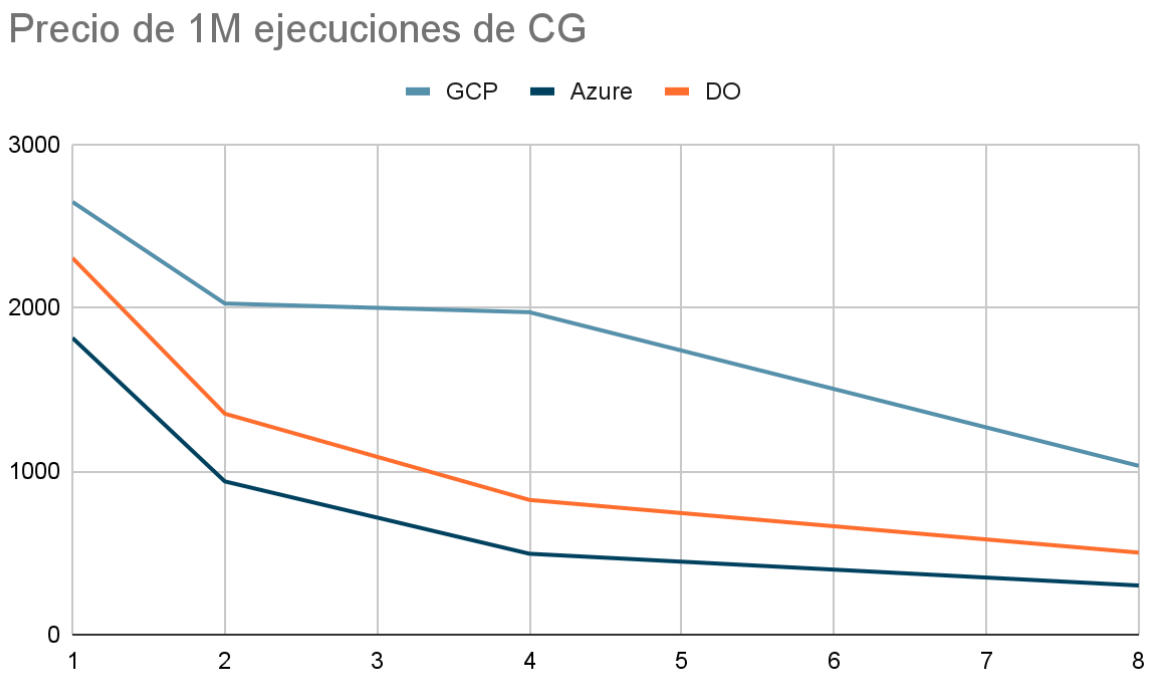




Estas gráficas son interesantes en tanto a que las eficiencias paralelas de DO y GCP son prácticamente opuestas. En el caso de la prueba CG observamos que GCP va incrementando su eficiencia entre las 4 y los 8 vCpus, llegando a posicionarse en la cima con la última configuración, mientras que DO comienza muy bien (superescalar en el caso del nodo local con dos cpus) y descendiendo de forma totalmente lineal. En el caso de LU vemos una jerarquía muy marcada por la dominancia de Azure.

4.3. Análisis de coste





En este gráfico sintético simulamos 1 millón de ejecuciones y observamos el precio total en USD. Si observamos detenidamente estas gráficas observamos que se trata de una copia casi exacta de las gráficas de tiempo en bruto, con esto observamos que la diferencia de precio de las instancias para un número importante de ejecuciones no es muy relevante y que el peor tiempo de ejecución de GCP redunda en un exceso de costes.

En general podemos concluir que DigitalOcean y Azure son prácticamente iguales en su idoneidad para ejecutar clústers HPC, mientras que GCP se queda en un claro tercer puesto, además introduciendo los resultados de AWS, observamos que estos son equivalentes (t3.medium) a los obtenidos con Azure y DO.

5. **Terraform, azure y dificultades**

Este es el apartado más diferencial de todo el trabajo, en este caso hemos dedicado un tiempo significativo a aprender a utilizar la herramienta Terraform (un agregado de unas 20 horas) para crear despliegues que pretendíamos fuesen agnósticos de la plataforma. Una vez indagado vemos que los planes de despliegue siguen siendo totalmente dependientes de la plataforma, incluso nos planteamos un proyecto de mejora por el cual desarrollemos un traductor de un lenguaje verdaderamente agnóstico a terraform.

Hemos realizamos multitud de modificaciones sobre el fichero CloudFormation de partida, por ejemplo: En el caso de Azure, no tenemos comandos de healthcheck, con lo que notificamos que el head node está listo a través de la creación de un fichero en el NFS, en DigitalOcean el virtualrouter no resuelve las DNS locales con lo que tenemos que crear un fichero de hosts manual en el NFS y enlazar el fichero de hosts local de la máquina a este.

La principal dificultad ha surgido en las exigencias para el uso del plugin de cgroups, un cgroup (control group) es un mecanismo que permite limitar el uso de CPU, memoria y operaciones de entrada/salida que puede realizar un conjunto de procesos. Cómo podemos imaginarnos slurm hace un uso exhaustivo de esta funcionalidad (en su configuración por defecto) sin embargo, la versión disponible en la compilación básica y la ofertada por el kernel no son compatibles a simple vista, con lo que debemos compilarla. Tras mucho investigar descubrimos que el script de makefile lo hará de forma automática si tiene las librerías de desarrollo de dbus (cosa curiosa porque en principio no tienen nada que ver, dbus es un subsistema para IPC). Además existe un bug en la versión de slurm comentado en su sitio web según el cual el servicio de systemd no podrá arrancar si no encuentra el fichero cgroup.conf (aunque este se encuentre vacío). Para solventarlo en la etapa de despliegue copiaremos el fichero de ejemplo a la carpeta $SLURM_HOME/etc/

Finalmente hemos encontrado lo que creemos que es un bug en la implementación original, la línea del yaml SlurmdSpoolDir=/var/spool/slurm/d debería ser SlurmdSpoolDir=/var/spool/slurm/ a secas (o para ser más correctos terminológicamente SlurmdSpoolDir=/var/spool/slurmd/).

6. **Cómo ejecutar:**

Esto se especifica en el fichero README.md, en general el procedimiento siempre es el mismo:

1. Obtener un token de autenticación válido para el proveedor cloud.
2. Ejecutar terraform init, terraform validate y terraform apply.
3. Agregar los exports a la sesión shell y ejecutar exec.sh
4. Copiar los datos desde /nfs/mpi/reports