FCCN UNIDADE DA FCT

Tecnologia para o Conhecimento

Memória descritiva sobre a instalação na Google Cloud Platform de sistema HPC de pequena escala

FCT-FCCN, Maio 2022

FCT Fundação para a Ciência e a Tecnologia

ÍNDICE

1. ENQUADRAMENTO

O cálculo científico recorre frequentemente à utilização de computação paralela em sistemas computacionais de alto desempenho, vulgo sistemas HPC – *High Performance Computing*.

A GCP – *Google Cloud Platform* – permite a instalação de sistemas HPC [1] que a presente memória descritiva testa simplificadamente, seguindo as instruções presentes no documento *Best practices for running tightly coupled HPC applications on Compute Engine*, disponível em https://cloud.google.com/architecture/best-practices-for-using-mpi-on-compute-engine, complementarmente a:

- Um conjunto de scripts auxiliares da Google, que pode ser encontrado em https://github.com/GoogleCloudPlatform/hpc-tools
- O documento *Creating an HPC-ready VM instance* que pode ser encontrado em https://cloud.google.com/compute/docs/instances/create-hpc-vm e com informação relevante para este teste.

A presente memória descritiva guiou-se por estas 3 fontes principais.

> Nota: este teste não constitui documentação sobre a utilização da GCP para qualquer propósito específico com efeitos produtivos, mas somente uma descrição dos passos seguidos para um teste de vários possíveis. Cada utilizador interessado deve procurar a forma de proceder que se ajusta melhor ao seu caso. Os descritivos procedimentais apresentados neste documento destinam-se a servir de prova de conceito, não sendo exaustivos, nem otimizados nem adotam uma política de segurança especifica. Para além da documentação disponível não foi solicitado suporte técnico da Google.
>
> Nota 2: os comandos descritos, caso sejam aproveitados, não podem ser usados diretamente, pois têm que ser adaptados a cada caso específico, adequando o nome de projeto GCP, zona na cloud, chave SSH, endereços IP, etc.

2. SOBRE A OFERTA HPC NA GCP

Através de recursos pré-configurados na GCP é possível lançar um *cluster* HPC até 660 v-CPU do tipo Intel Scalable Processor (Cascade Lake) ou AMD similar, com interligação dos nós de processamento via MPI. Outras configurações de maior dimensão poderão ser possíveis realizar, por exemplo recorrendo à oferta *bare metal*, embora não tenham sido analisadas essas alternativas para o presente documento.

Um cluster do tipo descrito anteriormente teria uma capacidade de 5,7 milhões de v-CPU.horas por ano, ou 2,9 core_CPU.horas por ano, dado que 2 v-CPU = 1 core-CPU. Para além de CPU

[1] https://cloud.google.com/solutions/hpc

genéricos, a GCP permite também aceder a processamento do tipo GPU, Tensor Flow e muitos outros tipos de processamento aplicacional.

3. PRÉ-REQUISITOS

É necessário ter uma conta ativa na GCP - https://console.cloud.google.com/- com os créditos necessários para criar os recursos como VMs, storage ou outros.

Deve ter uma chave SSH, para aplicação nas VMs criadas na GCP.

É aconselhável ter conhecimentos prévios de administração de sistemas operativos tipo Linux.

4. PREPARAÇÃO DA CONTA NA GCP

Na presente memória descritiva foi criado um novo acesso com a conta rnca.fccn@gmail.com, para aceder a https://console.cloud.google.com/.

Foi necessário associar à conta rnca.fccn@gmail.com um número de telefone (telemóvel) e um email alternativo.

Foi utilizado um código de *GCP Credits* previamente fornecido pela Google.

Porém, antes de poder colocar esse código, foi necessário criar uma conta de faturação, conforme mostrado na imagem seguinte.

O "Nº id fiscal da empresa (NIPC)" era um dado opcional pelo que não foi colocado, na criação da conta de faturação.

Foi necessário adicionar um número de cartão de crédito ou débito, ou outra forma de pagamento, tendo sido adicionado um cartão virtual de baixo valor, uma vez que para este teste não era necessário o consumo de qualquer verba.

Após adicionar a forma de pagamento referida anteriormente, já foi possível inserir o código promocional, conforme mostrado na imagem seguinte.

Após inserção do código promocional foi necessário proceder ao "upgrade e resgatar" como mostrado na imagem a seguir

Com a operação de "upgrade e resgatar" a conta deixou de ser uma mera conta de teste na GCP, passando a ser uma conta regular com menos limitações funcionais.

> NOTA: O utilizador da GCP deve ter presente os custos que a sua utilização implica, que podem ser substanciais. O portal GCP tem seções especificas para essas consultas e configurações especiais, como os *budget alerts* [2] ou outras

5. DESCRIÇÃO GERAL DO PROCESSO

Para a presente memória descritiva foram dados os seguintes passos:

1- Instalar o ambiente HPC

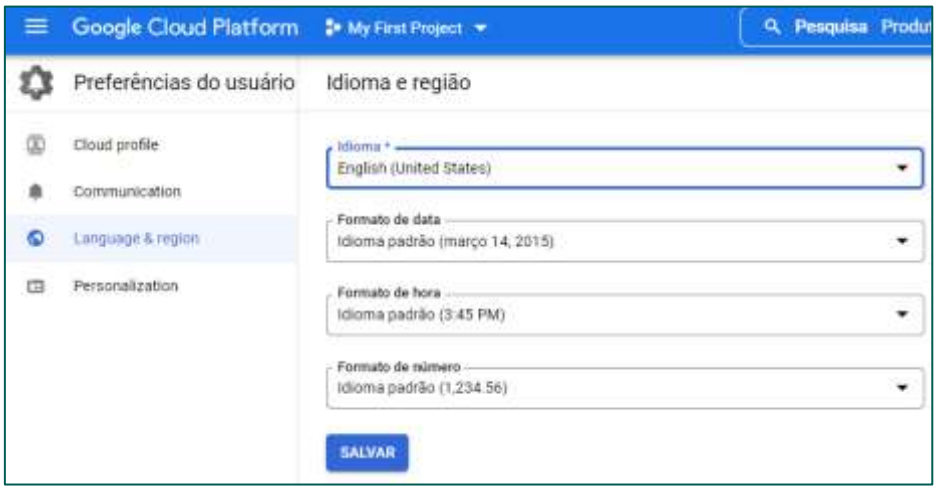
2- Correr uma aplicação de teste que recorra a comunicação inter-nodes em MPI

[2] https://cloud.google.com/billing/docs/how-to/budgets

3- Remover o ambiente HPC, que, se não for removido, consume recursos e pode gerar custos.

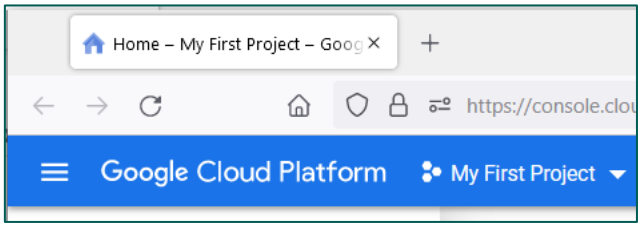
6. CRIAÇÃO DE UM PROJETO NA GCP

Para ser mais simples de seguir a documentação existente da Google, mudou-se a língua no portal GCP de Português para Inglês.



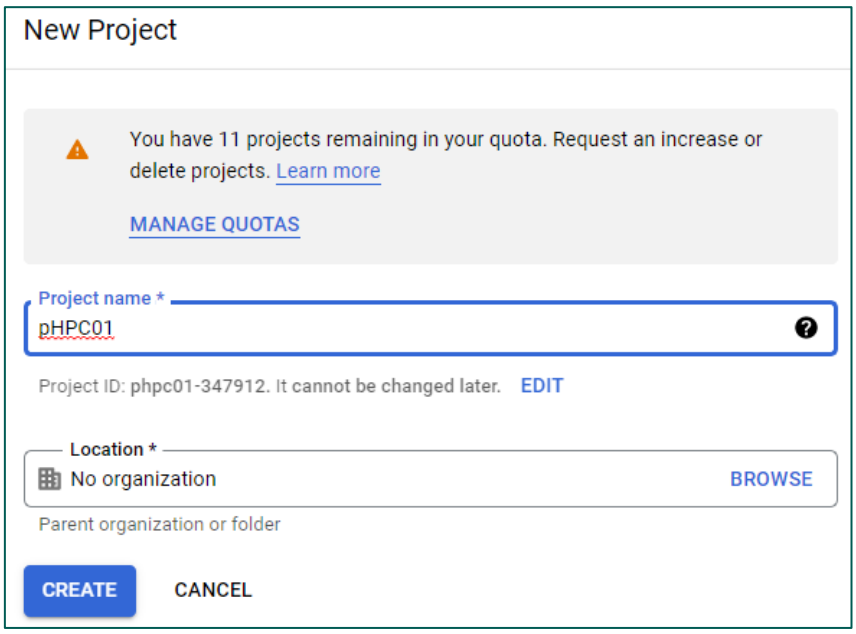
De seguida criou-se um projeto GCP.

Na consola da GCP - https://console.cloud.google.com – encontrar na barra de cima a localização do projeto de omissão. No caso vertente era “My First Project”.



A partir desse menu é possível criar um novo projeto. Neste caso foi criado o projeto pHPC01.

Novo projeto chamado “pHPC01”.



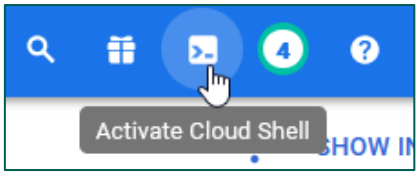
Após a criação do projeto, este foi selecionado para ficar como projeto ativo na consola GCP.

Sugere-se manter o "my first project" sem o apagar, para ficar como referência de configurações.

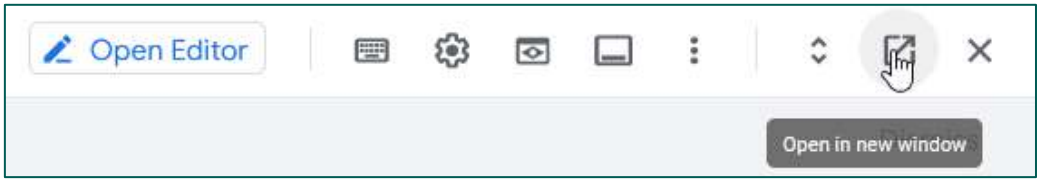
7. COMANDOS ATRAVÉS DO TERMINAL WEB DA GCP

Para dar comandos na GCP, é possível instalar uma CLI – *Command Line Interface* - no posto local ou então usar a CLI a partir do browser - https://console.cloud.google.com - através da barra de cima na opção Active Cloud Shell.

Os exemplos que se seguem usam esta segunda alternativa, não tendo sido necessário instalar software no posto local.



Depois de surgir a CLI pode escolher a opção *open in new Window* o que pode ser prático de usar, por dar mais espaço para o *output* dos comandos.



8. TESTAR A CRIAÇÃO DE UMA VM SIMPLES COM ACESSO POR SSH

É aconselhável testar a criação e acesso a uma VM simples na GCP antes de avançar, para se familiarizar com os procedimentos básicos da GCP.

Precisa ter, previamente, uma chave publica SSH para entrar na VM, que deve ser indicada na opção "--metadata=ssh-keys…". Neste exemplo usou-se o pacote de software putty, em SO Windows, para gerar e tratar a chave SSH.

Para a criação de uma VM simples, usámos a CLI a partir do browser. Pode dar os comandos seguintes.

Recomenda-se dar um comando de *export* de variável com a chave pública SSH, para reduzir o tamanho do comando *gcloud compute instances create*.

Notar bem que a variável "$chave_SSH" surge no parâmetro "--metadata=…".

O utilizador Linux onde essa chave SSH foi aplicada foi o "user01", ou seja, a GCP vai criar automaticamente um utilizador Linux (CentOS) chamado "user01" com entrada de rede com chave SSH.

```
$ export chave_SSH="ssh-rsa ……. user01@fccn.pt"

 $ gcloud compute instances create hpc-vm01 --zone=europe-
west1-b --image-family=hpc-centos-7 --image-
project=cloud-hpc-image-public --machine-type=e2-micro -
-metadata=ssh-keys=user01:"$chave_SSH" --project phpc01-347912

(teve o seguinte output)

API [compute.googleapis.com] not enabled on project […]. Would you
like to enable and retry (this will take a few minutes)? (y/N)? y

Enabling service [compute.googleapis.com] on project [….]...
Operation "operations/acf.p2-……. " finished successfully.
Created [https://www.googleapis.com/compute/v1/projects/phpc01-
347912/zones/europe-west1-b/instances/hpc-vm01].
NAME: hpc-vm01
ZONE: europe-west1-b
MACHINE_TYPE: e2-micro
PREEMPTIBLE:
INTERNAL_IP: 10.132.0.2
EXTERNAL_IP: 35.205.35.248
STATUS: RUNNING
```

Explicação dos comandos:

- "gcloud compute instances create" = operação que está a ser invocada
- " hpc-vm01" = nome da VM a criar
- "--zone=europe-west1-b" = zona geográfica na cloud da Google

- "--image-family=hpc-centos-7" = família da imagem
- "--image-project=cloud-hpc-image-public" = nome da image
- "--machine-type=e2-micro" = capacidade da VM
- "--metadata=ssh-keys=user01:"$chave_SSH"" = chave SSH a usar
    - NOTAR: a chave foi exportada previamente para a variável "chave_SSH"
- "--project phpc01-347912" = projecto no GCP. Notar que o nome do projeto não é "pHPC01" mas sim o ID que surge na "prompt" da Shell
    - NOTA: pode correr na CLI o comando "$ gcloud projects list" para obter a lista de projetos na GCP

Para verificar as características da VM criada, correu-se o comando seguinte:

```
$ gcloud compute instances describe  hpc-vm01 --zone=europe-west1-b
```

Poderá depois entrar por SSH [3] em user01@<endereço IP> para testar a VM, por exemplo:

```
(comando de teste de conetividade a dar na VM criada pela GCP)

[user01@hpc-vm01 ~]$ ping -c 2 www.google.com
PING www.google.com (108.177.119.147) 56(84) bytes of data.
64 bytes from ei-in-f147.1e100.net (108.177.119.147): icmp_seq=1 ttl=115 time=0.837 ms
64 bytes from ei-in-f147.1e100.net (108.177.119.147): icmp_seq=2 ttl=115 time=1.23 ms

--- www.google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 0.837/1.037/1.238/0.203 ms
```

A lista de regiões e zonas pode ser encontrada em
https://cloud.google.com/compute/docs/regions-zones

A zona europe-west1-b na Bélgica está classificada com de baixas emissões de CO2 [4].

europe-west1-b	St. Ghislain, Belgium, Europe	E2, N2, N2D, T2D, N1, M1, M2, C2	Ivy Bridge, Sandy Bridge, Haswell, Broadwell, Skylake, Cascade Lake, AMD EPYC Rome, AMD EPYC Milan	GPUs	Low CO_2

Sendo o sistema operativo do tipo "centos", dado que foi criada com "--image-family=hpc-centos-7", então deve fazer updates de software com um comando do tipo:

```
$ sudo yum update -y
```

No fim deste teste poderá apagar a VM:

```
$ gcloud compute instances stop hpc-vm01 --zone=europe-west1-b --project phpc01-347912
$ gcloud compute instances delete  hpc-vm01 --zone=europe-west1-b --project phpc01-347912
```

[3] O cliente SSH tem que ter a chave privada SSH que combina com a chave pública
[4] Infelizmente não foi possível usar nos testes de 4 VMs devido a problemas de capacidade

Nota: Se não apagar a VM pode ficar a consumir recursos na GCP, com expressão no processo de faturação.

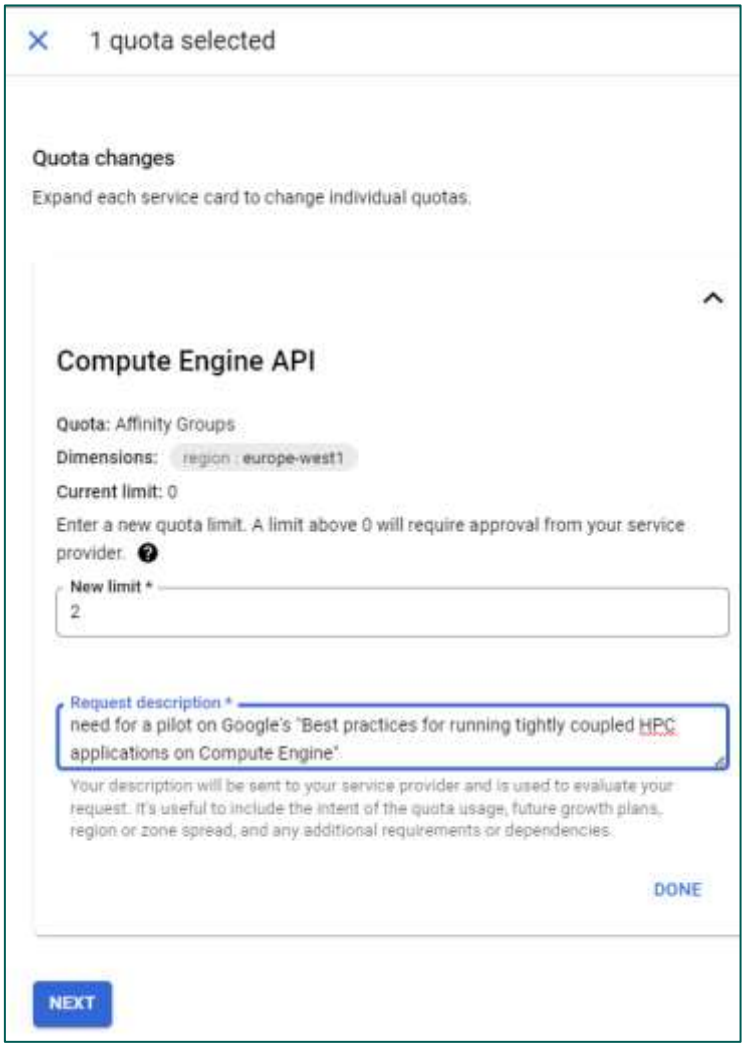
9. CRIAÇÃO DE VMS PARA HPC

Para este exemplo foram criadas 4 VMs iguais, seguindo o documento *Best practices for running tightly coupled HPC applications on Compute Engine*. De acordo com a documentação podem ser criadas até 11 VMs deste tipo.

Quota 'AFFINITY_GROUPS'

Antes de criar a regra de *group-placement* das VMs é necessário aumentar a quota de 'AFFINITY_GROUPS', caso contrário o comando de criação da regra resulta num erro do tipo *"Quota 'AFFINITY_GROUPS' exceeded. Limit: 0.0 in region…"*.

Para isso localizou-se a quota em questão no portal de gestão GCP e fez-se um pedido de aumento de quota. O pedido é feito para uma região da GCP. Cada região pode conter várias zonas.



Neste exemplo o pedido foi atendido em menos de uma hora, vindo a resposta por email.

O comando para criar a regra de *group-placement* foi o seguinte:

```
$ gcloud compute resource-policies create group-placement pp-hpc01 --collocation=collocated --vm-count=4 --project phpc01-347912 --region=europe-west1
```

A descrição das *machine types* C2 pode ser encontrada em https://cloud.google.com/compute/docs/compute-optimized-machines#c2_machine_types

Quota 'CPU'

Da mesma forma que se fez para a Quota 'AFFINITY_GROUPS' foi feito um pedido para upgrade de quota de C2-CPUs, neste caso para 300 v-CPU. Este pedido demorou mais de um dia a processar, tendo sido objeto de validação comercial.

Após aumento de quota foram criadas as VMs através da CLI do Portal da GCP.

Comando para criação das VMs:

```
 $ gcloud compute instances create    hpc11 hpc12 hpc13 hpc14  \
   --zone=europe-west1-c  \
  --resource-policies=pp-hpc01  \
 --no-restart-on-failure  \
  --maintenance-policy=TERMINATE  \
  --project phpc01-347912 \
 --machine-type=c2-standard-60 \
--image-family=hpc-centos-7  \
--image-project=cloud-hpc-image-public \
--network-interface=nic-type=GVNIC \
 --metadata=ssh-keys=user01:"$chave_SSH"
```

(teve como *output*)

```
Created [https://www.googleapis.com/compute/v1/projects/phpc01-347912/zones/europe-west1-c/instances/hpc11].
Created [https://www.googleapis.com/compute/v1/projects/phpc01-347912/zones/europe-west1-c/instances/hpc12].
Created [https://www.googleapis.com/compute/v1/projects/phpc01-347912/zones/europe-west1-c/instances/hpc13].
Created [https://www.googleapis.com/compute/v1/projects/phpc01-347912/zones/europe-west1-c/instances/hpc14].
NAME: hpc11
ZONE: europe-west1-c
MACHINE_TYPE: c2-standard-60
PREEMPTIBLE:
INTERNAL_IP: 10.132.0.50
EXTERNAL_IP: 34.77.113.239
STATUS: RUNNING
```

```
NAME: hpc12
ZONE: europe-west1-c
MACHINE_TYPE: c2-standard-60
PREEMPTIBLE:
INTERNAL_IP: 10.132.0.49
EXTERNAL_IP: 34.140.246.159
STATUS: RUNNING

NAME: hpc13
ZONE: europe-west1-c
MACHINE_TYPE: c2-standard-60
PREEMPTIBLE:
INTERNAL_IP: 10.132.0.51
EXTERNAL_IP: 34.79.141.150
STATUS: RUNNING

NAME: hpc14
ZONE: europe-west1-c
MACHINE_TYPE: c2-standard-60
PREEMPTIBLE:
INTERNAL_IP: 10.132.0.52
EXTERNAL_IP: 35.241.242.13
STATUS: RUNNING
```

Este comando foi expandido e adaptado da documentação do Google.

Explicação dos parâmetros:

- “hpc11 hpc12 hpc13 hpc14 \”=
  - nome das Vms a criar
- “ --zone=europe-west1-c \” =
  - Zona na GCP. Não foi possível usar a zona “-b” por limitações pontuais de capacidade naquela zona
  - NOTA: a região “europe-west1” estaria com capacidade limitada na altura dos testes, pois nem sempre foi possível lançar esta infraestrutura de 4 Vms de alta capacidade, dando um erro temporário de "*The zone '...../europe-west1…' does not have enough resources available to fulfill the request.* ". Outras regiões poderiam estar mais folgadas, mas o assunto não foi investigado mais aprofundadamente.
- “ --resource-policies=pp-hpc01 \” =
  - instanciação de política de colocação criada anteriormente, que garante que as VMs serão colocadas próximas umas das outras, o que é essencial para um cluster HPC
- “ --no-restart-on-failure \” =
  - no caso de manutenção de hipervisora a GCP não tem que relançar a VM

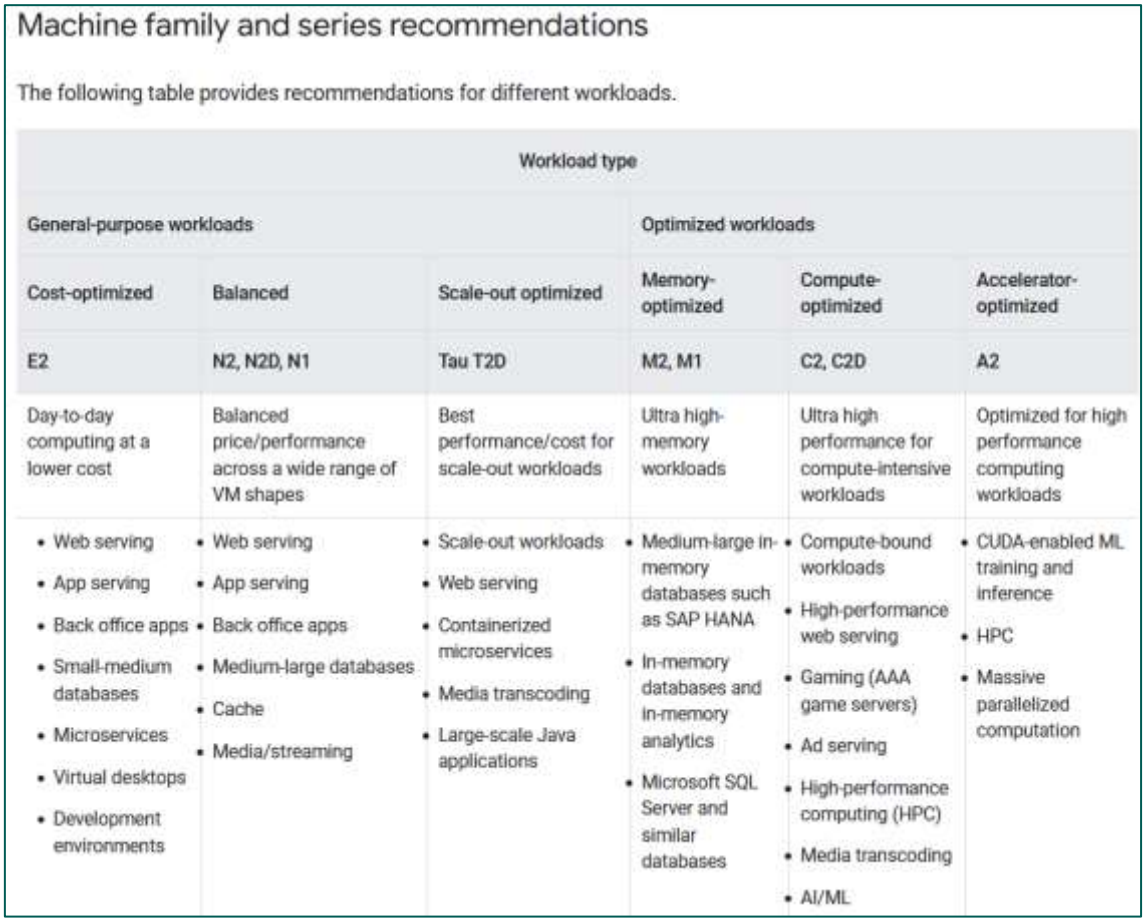
- “ --maintenance-policy=TERMINATE \” =
  - no caso de manutenção de hipervisora a GCP faz shutdown da VM
- “ --project phpc01-347912 \” =
  - identificação do projeto
- “ --machine-type=c2-standard-60 \” =
  - arquitetura da VM que são 60 v-CPU
- “--image-family=hpc-centos-7 \” =
  - tipo de Sistema operativo
- “--image-project=cloud-hpc-image-public \” =
  - imagem a usar para a VM
- “--network-interface=nic-type=GVNIC \” =
  - porta de rede especial otimizada
- “ --metadata=ssh-keys=user01:"$chave_SSH" “=
  - chave publica SSH a aplicar no utilizador Unix “user01”

NOTA: este tipo de VM com 60 v-CPU tem um custo substancial. A tabela seguinte permite ter uma aproximação, por baixo, dos custos mensais para cada VM deste tipo. As VMs tipo C2 serão mais dispendiosas do que as T2.

Belgium (europe-west1) Monthly

Machine type	Virtual CPUs	Memory	Price (USD)	Spot price* (USD)
t2d-standard-1	1	4GB	$33.93	$8.21
t2d-standard-2	2	8GB	$67.85	$16.42
t2d-standard-4	4	16GB	$135.7	$32.84
t2d-standard-8	8	32GB	$271.4	$65.67
t2d-standard-16	16	64GB	$542.8	$131.34
t2d-standard-32	32	128GB	$1085.61	$262.68
t2d-standard-48	48	192GB	$1628.41	$394.02
t2d-standard-60	60	240GB	$2035.52	$492.53

A tabela seguinte – disponível em https://cloud.google.com/compute/docs/machine-types - resume as famílias de VMs disponíveis.



Machine family and series recommendations

The following table provides recommendations for different workloads.

Workload type					
General-purpose workloads			Optimized workloads		
Cost-optimized	Balanced	Scale-out optimized	Memory-optimized	Compute-optimized	Accelerator-optimized
E2	N2, N2D, N1	Tau T2D	M2, M1	C2, C2D	A2
Day-to-day computing at a lower cost	Balanced price/performance across a wide range of VM shapes	Best performance/cost for scale-out workloads	Ultra high-memory workloads	Ultra high performance for compute-intensive workloads	Optimized for high performance computing workloads
• Web serving • App serving • Back office apps • Small-medium databases • Microservices • Virtual desktops • Development environments	• Web serving • App serving • Back office apps • Medium-large databases • Cache • Media/streaming	• Scale-out workloads • Web serving • Containerized microservices • Media transcoding • Large-scale Java applications	• Medium-large in-memory databases such as SAP HANA • In-memory databases and in-memory analytics • Microsoft SQL Server and similar databases	• Compute-bound workloads • High-performance web serving • Gaming (AAA game servers) • Ad serving • High-performance computing (HPC) • Media transcoding • AI/ML	• CUDA-enabled ML training and inference • HPC • Massive parallelized computation

Cada VM foi criada com 60 v-CPUs do tipo "Intel(R) Xeon(R) CPU @ 3.10GHz" cujo detalhe pode ser consultado no ficheiro /proc/cpuinfo [5] . Ou seja, cada VM tem disponíveis 60 v-CPU deste tipo.

10. AMBIENTE DE CLUSTER

Neste exemplo **o servidor "11" (hpc11) foi convencionado de "controller" do cluster**, pelo que os comandos no cluster são executados a partir de lá.

Apenas o servidor hpc11 precisaria de endereço IP público, embora todas as instâncias tenham um endereço IP público atribuído automaticamente pela GCP.

10.1 Criação de ficheiro de "hosts"

Na consola da GCP foi indicado o projeto para evitar ter que informar esse ponto em cada comando de "gcloud":

[5] Após aplicar a configuração *Disable Hyper-Threading* e fazer reboot, os 60 CPU passam a 30 em /proc/cpuinfo, passando a haver uma correspondência 1:1 entre v-CPU e core-CPU, o que é uma configuração recomendável para HPC, embora possam ser feitos testes aplicacionais para aferir, por aplicação, qual a configuração mais otimizada (60 v-CPU versus 30 v-CPU).

```
$ gcloud config set project phpc01-347912
```

Na consola da GCP obteve-se os endereços IP das VMs a correr com o comando seguinte:

```
$ gcloud compute instances list|egrep 'NAME|_IP'
(teve o output seguinte)
…
NAME: hpc11
INTERNAL_IP: 10.132.0.50
EXTERNAL_IP: 35.241.242.13
NAME: hpc12
INTERNAL_IP: 10.132.0.49
EXTERNAL_IP: 34.77.113.239
NAME: hpc13
INTERNAL_IP: 10.132.0.51
EXTERNAL_IP: 34.140.246.159
NAME: hpc14
INTERNAL_IP: 10.132.0.52
EXTERNAL_IP: 34.79.141.150
```

Foram mapeados os endereços IP **privados** do cluster no ficheiro /etc/hosts.

A informação revelante dos endereços foi criada na ficheiro /etc/hosts do controller do cluster:

```
…..
################## enderecos internos do cluster:
10.132.0.50 hpc11
10.132.0.49 hpc12
10.132.0.51 hpc13
10.132.0.52 hpc14
……….
```

NOTA: Estes endereços internos são mantidos entre STOP/START das VMs, embora os endereços IP públicos mudem entre cada uma dessas operações.

NOTA 2: Caso os discos sejam lançados novamente em VMs novas, os endereços IP privados mudam também

AVISO: Caso se faça delete da VM e o seu disco não for persistente, então perde-se o disco da VM. Antes de fazer delete pode fazer STOP da VM e fazer um clone do disco.

10.2 Teste conetividade intra-cluster

Com o ficheiro /etc/hosts devidamente preenchido fez-se um teste de conetividade a partir do controller, através do comando seguinte:

```
[user01@hpc11 ~]$ for i in 1 2 3 4 ; do ping -c 1 hpc1$i |grep loss ;
done
(teve o output seguinte)
1 packets transmitted, 1 received, 0% packet loss, time 0ms
1 packets transmitted, 1 received, 0% packet loss, time 0ms
```

```
1 packets transmitted, 1 received, 0% packet loss, time 0ms
1 packets transmitted, 1 received, 0% packet loss, time 0ms
```

Notar que o *round trip time* dá 0 mili-segundos o que é algo positivo e indicador que as VMs estarão próximas entre elas do ponto de vista de rede.

10.3 Copiar chave privada para o Controller do cluster (HPC-vm01)

O utilizador Unix "user01" tem pré-configurada a chave SSH usada na criação de cada VM tendo a capacidade "sudo" sem password, pois, como referido anteriormente, a GCP colocou automaticamente essa chave pública nas VMs que criou.

Basta agora colocar a **chave privada** correspondente no utilizador Unix "user01" do controller, para que este servidor possa entrar por SSH em todo o cluster. O ficheiro foi o /home/user01/.ssh/id_rsa.

Exemplo de um ficheiro /home/user01/.ssh/id_rsa :

```
-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQEAvQkB3MBqmVNE/VOFZkKrfrFo/DPGbBHF…….
…….. (SNIP) ……
MknSaa0jiAA7ZuYjudye7XLwg3XlUrO00exgIcsItf2wadMj0Q==
-----END RSA PRIVATE KEY-----
```

Notar bem as permissões 600 do ficheiro:

```
 $ ls -l .ssh/id_rsa
-rw------- 1 user01 user01 1675 Apr 22 15:34 .ssh/id_rsa
```

10.4 Teste da entrada SSH no cluster a partir do Controller

A seguir testou-se a entrada a partir do servidor controller em todo o cluster:

- Da primeira vez que corre o comando seguinte, pede para confirmar a chave SSH de cada nó do cluster o que deve ser feito.
- Da vezes subsequentes já não pede essa confirmação.
- Comando dado a partir do controller do cluster:

```
[user01@hpc11 ~]$ for i in 1 2 3 4 ; do ssh hpc1$i 'uname -a;date' ; done

 …..
```

10.5 Distribuir ficheiro de hosts por todo o cluster

A seguir copiou-se o ficheiro hosts para todo o cluster.

```
[user01@hpc11 ~]$ for i in 1 2 3 4 ; do scp /etc/hosts hpc1$i: ; ssh hpc1$i
"sudo cp hosts /etc/hosts" ; done
```

10.6 Entradas SSH automáticas

O comando mpirun vai precisar de utilizar o SSH automaticamente pelo que não deve bloquear na verificação do checksum de host SSH.

Para evitar isso é possível distribuir os "checksum de host SSH" em todo o cluster, como se fez para o ficheiro de hosts ou então configurar o SSH para não fazer a verificação.

Neste exemplo optou-se pela segunda alternativa.

> NOTA: numa configuração de produção, com segurança mais controlada, pode ser mais aconselhável distribuir um ficheiro de checksum SSH e manter essa verificação.

O ficheiro ~/.ssh/config teve o seguinte conteúdo, para a não verificação de checksum SSH:

```
Host *
StrictHostKeyChecking no
```

O ficheiro ~/.ssh/config tem as permissões 600.

A partir do "hpc-vm01" correu-se o comando seguinte para distribuir no cluster o ficheiro ~/.ssh/config.

```
[user01@hpc11 ~]$ for i in 1 2 3 4  ; do scp ~/.ssh/config hpc1$i:.ssh/config  ; done
```

10.7 Ajuste do fuso horário

A partir do controller ajustou-se o fuso horário no cluster todo:

```
[user01@hpc11 ~]$ for i in 1 2 3 4 ; do  ssh hpc1$i "sudo timedatectl set-timezone Europe/Lisbon;date"; done
```

10.8 Verificação do serviço NTP

A partir do controller verificou-se que estava a correr o acerto automático de relógio pela rede (NTP).

> NOTA: é recomendável que todos os nós dos clusters tenham a mesma referência de relógio.

```
[user01@hpc11 ~]$ for i in 1 2 3 4 ; do ssh hpc1$i "sudo  systemctl status chronyd| grep Active"; done
(teve o output seguinte)
  Active: active (running) since Thu 2022-04-28 16:29:10 WEST; 22min ago
  Active: active (running) since Thu 2022-04-28 16:28:29 WEST; 23min ago
  Active: active (running) since Thu 2022-04-28 16:28:31 WEST; 23min ago
  Active: active (running) since Thu 2022-04-28 16:29:07 WEST; 22min ago
```

Se o NTP não estiver a correr, ativar esse serviço.

10.9 Instalação do parallel ssh

Para alguns comandos mais demorados é preferível corrê-los, não como um ciclo "for" que serializa os comandos em cada nó do cluster, mas sim em paralelo, ao mesmo tempo em todo o cluster.

Assim instalou-se o pssh no controller:

```
[user01@hpc11 ~]$ sudo yum -y install pssh

---------------------------------------- Foi criado um ficheiro hosts.pssh  com a seguinte lista:
hpc11
hpc12
hpc13
hpc14
```

(notar que a lista de cima corresponde ao conteúdo do ficheiro /etc/hosts)

---------- Teste:
[user01@hpc11 ~]$ pssh -i -h hosts.pssh uptime

(teve o output seguinte)

```
[1] 16:55:49 [SUCCESS] hpc11
 16:55:49 up 26 min,  1 user,  load average: 0.01, 0.02, 0.03
[2] 16:55:49 [SUCCESS] hpc12
 16:55:49 up 27 min,  0 users,  load average: 0.00, 0.01, 0.03
[3] 16:55:49 [SUCCESS] hpc13
 16:55:49 up 27 min,  0 users,  load average: 0.00, 0.01, 0.02
[4] 16:55:49 [SUCCESS] hpc14
 16:55:49 up 26 min,  0 users,  load average: 0.00, 0.01, 0.03
```

10.10 Upgrade de software via pssh / modernização do software instalado

A partir do controller correu-se o comando seguinte, que demorou vários minutos a correr, pelo que tem o parâmetro "-t 3600", que permite demorar até uma hora.

> NOTA: deve fazer-se um update de software a seguir a uma instalação fresca para remendar vulnerabilidades de segurança ou resolver outro tipo de problemas.

```
[user01@hpc11 ~]$ pssh -h hosts.pssh    -t 3600 -- sudo yum -y
upgrade
```

De seguida foi realizado um reboot de todo o cluster a partir do controller.

> NOTA: notar bem a ordem do ciclo "for" que está propositadamente invertida, pois o último a fazer reboot tem que ser o controller, o servidor #1.

```
[user01@hpc11 ~]$ for i in 4 3 2 1   ; do ssh hpc1$i "sudo  reboot "; done
```

11. INSTALAÇÃO DAS FERRAMENTAS HPC DA GOOGLE (SCRIPTS)

Foi instalado no cluster o pacote de scripts da Google disponível em https://github.com/GoogleCloudPlatform/hpc-tools

A partir do controller correu-se o comando seguinte:

```
[user01@hpc11 ~]$ pssh -h hosts.pssh -- git clone https://github.com/GoogleCloudPlatform/hpc-tools
```

Que instalou uma serie de utilitários, nomeadamente os seguintes:

```
[user01@hpc11 ~]$ ls hpc-tools/
contributing.md                google_install_mpi       mpi-tuning.sh
google_hpc_firstrun            google_install_mpitune   packaging
google-hpc-firstrun.service    LICENSE                  README.md
google_hpc_multiqueue          mpitune-configs          vmroot
google-hpc-multiqueue.service  mpi-tuning-ansible.yaml
[user01@hpc11 ~]$
```

O ficheiro de readme que acompanha os scripts tem informação sobre a utilidade de cada ficheiro: https://github.com/GoogleCloudPlatform/hpc-tools/blob/master/README.md

Algumas configurações disponíveis:

1. tcpmem - Increase memory for TCP
2. networklatency - Enable busy polling and low network latency tuned profile
3. limits - Change the system ulimits
4. nosmt - Disable simultaneous multi threading
5. nofirewalld - Disable firewalld
6. noselinux - Disable SE Linux
7. nomitigation - Disable CPU vulnerabilities mitigations
8. hpcprofile - (Bash only) Apply the tuned profile including the following: tcpmem, networklatency (busypoll), nosmt, noselinux

Dado que estas configurações eram todas relevantes, foram todas aplicadas com o comando seguinte:

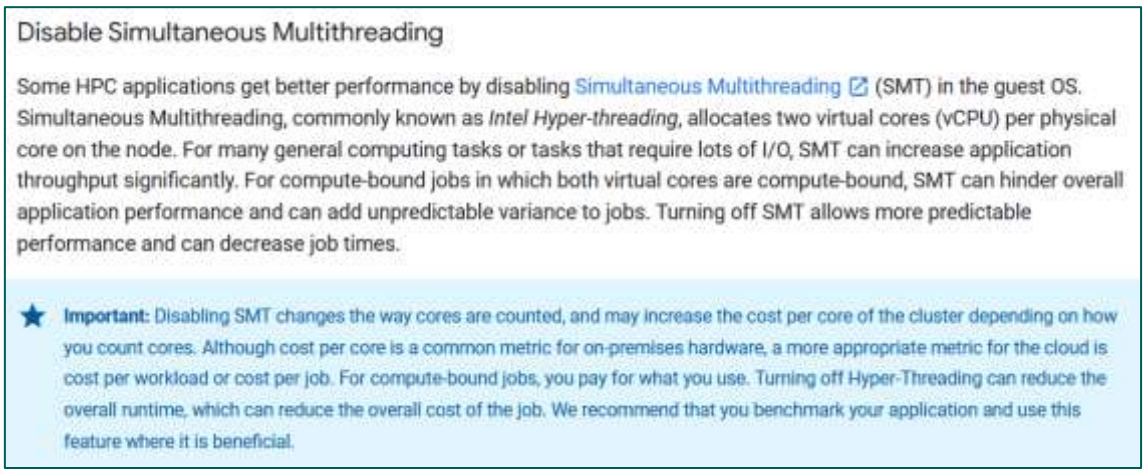
```
[user01@hpc11 ~]$ for i in tcpmem networklatency limits nosmt nofirewalld noselinux nomitigation hpcprofile ; do echo $i ; pssh -h hosts.pssh -- sudo hpc-tools/mpi-tuning.sh --${i} ; done
```

De seguida foi realizado um reboot de todo o cluster a partir do controller.

NOTA: notar bem a ordem do ciclo “for” que está propositadamente invertida

```
[user01@hpc11 ~]$ for i in 4 3 2 1 ; do ssh hpc1$i "sudo reboot "; done
```

NOTA: após a configuração “nosmt - Disable simultaneous multi threading” tomar efeito, o Sistema Operativo passa a reportar 30 CPUs em vez de 60 em cada nó. Notar bem a seguinte nota especial da documentação:



Disable Simultaneous Multithreading

Some HPC applications get better performance by disabling Simultaneous Multithreading (SMT) in the guest OS. Simultaneous Multithreading, commonly known as *Intel Hyper-threading*, allocates two virtual cores (vCPU) per physical core on the node. For many general computing tasks or tasks that require lots of I/O, SMT can increase application throughput significantly. For compute-bound jobs in which both virtual cores are compute-bound, SMT can hinder overall application performance and can add unpredictable variance to jobs. Turning off SMT allows more predictable performance and can decrease job times.

Important: Disabling SMT changes the way cores are counted, and may increase the cost per core of the cluster depending on how you count cores. Although cost per core is a common metric for on-premises hardware, a more appropriate metric for the cloud is cost per workload or cost per job. For compute-bound jobs, you pay for what you use. Turning off Hyper-Threading can reduce the overall runtime, which can reduce the overall cost of the job. We recommend that you benchmark your application and use this feature where it is beneficial.

NOTA 2: a configuração de "Disable simultaneous multi threading" pode ser feita através do script seguido de reboot, tal como foi feita neste teste, ou então através de comandos GCP na altura de lançamento das VMs. Como estava disponível um script, optou-se por esta via.

Verificações sobre a aplicação dos scripts.

Foram feitas algumas verificações no cluster que, de facto, os scripts tiveram o efeito esperado.

Para verificar o estado do selinux no cluster, deu-se o comando seguinte:

```
[user01@hpc11 ~]$ for i in 1 2 3 4 ; do ssh hpc1$i "sudo getenforce"; done
(teve o output seguinte)

Disabled
Disabled
Disabled
Disabled
```

Para verificar o estado do iptables no cluster, deu-se o comando seguinte:

```
[user01@hpc11 ~]$ for i in 1 2 3 4 ; do ssh hpc1$i "sudo iptables -L"; done

(teve o output seguinte)
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
……..
```

12. INSTALAÇÃO DE ARMAZENAMENTO DE DADOS PARTILHADO (STORAGE)

Tipicamente a aplicação de HPC corre em vários nós em paralelo e precisa de aceder a um sistema de armazenamento de dados partilhado. No mínimo isso é aconselhável para arrancar o binário da aplicação HPC que deve ser o mesmo em todos os nós do cluster [6]. Mais usualmente, o *storage* partilhado é usado para guardar os resultados das *runs* para *checkpointing* [7] aplicacional, ou como parte integrante do processamento, com trocas frequentes de dados entre RAM e storage, para partilha de informação massiva entre os nós do cluster.

Em sistemas de HPC de larga escala utiliza-se normalmente soluções de sistemas de ficheiros de acesso paralelo, de alta performance, muitas vezes ligados diretamente à rede de alto-débito de fibra ótica, ou seja, a mesma rede que interliga os microprocessadores dos nós computacionais.

A documentação da Google *Best practices for running tightly coupled HPC applications on Compute Engine* refere dois tipos possíveis na GCP disponíveis para instalar:

- NFS-based solutions such as Filestore and NetApp Cloud Volumes are the easiest for deploying shared storage options. Both options are fully managed on Google Cloud, and are best when the application doesn't have extreme I/O requirements to a single dataset, and has limited to no data sharing between compute nodes during application execution and updates. For performance limits, see the Filestore and NetApp Cloud Volumes documentation.
- POSIX-based parallel file systems are more commonly used by MPI applications. POSIX-based options include open source Lustre and the fully supported Lustre offering, DDN Storage EXAScaler Cloud. When compute nodes generate and share data, they frequently rely on the extreme performance provided by parallel file systems and support for full POSIX semantics. Parallel file systems like Lustre deliver data to the largest supercomputers and can support thousands of clients. Lustre also supports data and I/O libraries such as NetCDF and HDF5, along with MPI-IO, enabling parallel I/O for a wide set of application domains.

No presente teste utilizou-se uma solução mais simples, que foi instalar um servidor NFS configurado num dos servidores do *cluster* de 4 nós.

> NOTA: numa instalação de produção com requisitos de alto desempenho, esta solução não é aconselhável. Em vez disso, considerar antes as opções indicadas na documentação da Google.

[6] Também poderá ser possível copiar o binário para disco local de cada nó, mas essa configuração não é muito habitual, sendo muito mais habitual usar algum tipo de storage partilhado.
[7] Que permita o re-arranque da aplicação paralela no caso de ter sido interrompida por qualquer razão.

NOTA 2: mesmo numa instalação de produção sem requisitos de alto desempenho, não é conveniente misturar funções de servidor de storage com funções de compute, pois isso pode desequilibrar e prejudicar o processamento de todo o cluster.

Encontrou-se documentação para instalar o Lustre *em Introducing Lustre file system Cloud Deployment Manager scripts* - https://cloud.google.com/blog/products/storage-data-transfer/introducing-lustre-file-system-cloud-deployment-manager-scripts

Também há documentação para outras soluções possíveis no portal da GCP.

12.1 Instalação de um servidor NFS

Dado que o objetivo do presente teste não teve requisitos de alto desempenho de storage, optou-se por instalar um servidor NFS no servidor Controller para simplificação do processo.

A partir do controller correu-se o comando seguinte para instalar o software de servidor NFS.

```
$ sudo yum -y install nfs-utils nfs-utils-lib
```

NOTA: as políticas de firewalls de rede da GCP permitem apenas tráfego de entrada de ICMP e SSH, pelo que estes serviços não ficam expostos à Internet.

A partir do controller correu-se os comandos seguinte, para ativar o serviço NFS

```
$ sudo chkconfig nfs on
$ sudo chkconfig nfslock on
$ sudo service rpcbind start
$ sudo service nfs start
$ sudo service nfslock start
```

No controller criou-se um diretório de exportação para NFS:

```
[user01@hpc11 ~]$ sudo mkdir -p  /nfs_export/user01
[user01@hpc11 ~]$ sudo chown user01:user01  /nfs_export/user01
```

De seguida editou-se o ficheiro /etc/exports para configuração do servidor NFS, tendo ficado com esta configuração:

```
/nfs_export 10.132.0.0/24(rw,sync,no_root_squash,no_subtree_check,insecure)
```

Esta configuração exporta o “share” NFS “nfs_export” com uma série de opções.

O share será montado no cluster na pasta local “/hpc”.

O share foi exportado:

```
[user01@hpc11 ~]$ sudo  exportfs -a
```

O share foi testado localmente no controller:

```
[user01@hpc11 ~]$ sudo mount 10.132.0.50:/nfs_export  /hpc
[user01@hpc11 ~]$ sudo umount    /hpc
```

A partir do controller o share foi montado em todo o cluster através do comando:

```
$ for i in 1 2 3 4    ; do   ssh hpc1$i "sudo mkdir /hpc ; \
sudo chown user01:user01  /hpc ; \
 sudo  mount 10.132.0.50:/nfs_export  /hpc ; \
```

```
"   ; done
```

A partir do controller o share foi testado em todo o cluster através do comando:

```
$ for i in 1 2 3 4    ; do   ssh hpc1$i " \
 sudo df -h |grep /hpc  ; \
 "   ; done
```

(que teve o output seguinte)

```
10.132.0.50:/nfs_export   20G  3.1G   17G  16% /hpc
10.132.0.50:/nfs_export   20G  3.1G   17G  16% /hpc
10.132.0.50:/nfs_export   20G  3.1G   17G  16% /hpc
10.132.0.50:/nfs_export   20G  3.1G   17G  16% /hpc
```

IMPORTANTE: Sempre que o cluster arranque, será necessário montar o share NFS como feito atrás.

Alternativamente o share pode ficar indicado no ficheiro /etc/fstab. Para tornar a configuração permanente pode adicionar-se a linha "10.132.0.50:/nfs_export /hpc nfs defaults 0 0" ao ficheiro /etc/fstab. No presente teste não se considerou necessária essa configuração permanente.

AVISO: no caso de ausência de mount, então cada servidor do cluster fica a aceder localmente à pasta /hpc em vez de aceder remotamente ao servidor hpc-vm01, como é o objetivo pretendido. Esta situação, se ocorrer, torna-se confusa do ponto de vista operacional.

13. INSTALAÇÃO DO MPI

A documentação da Google *Best practices for running tightly coupled HPC applications on Compute Engine* recomenda:

"*…For best performance, we recommend that you use Intel MPI"*

Documentação adicional [8] refere que:

"*…Use google_install_mpi to install IntelMPI environment on indvidual VMs."*

Documentação de ajuda do script *google_install_mpi* refere o seguinte:

```
Usage:
  Verify installation steps: google_install_mpi [options] --dryrun
  Apply installation: google_install_mpi [options]

Options:
  -h | --help        Display help message
```

[8] https://github.com/GoogleCloudPlatform/hpc-tools/blob/master/README.md

```
--dryrun          Do not execute commands
--prefix          Configure the prefix directory for installations
                  Default location is set to /opt/intel
--intel_checker   Install Intel(R) Cluster Checker
--intel_compliance  Configure environment in compliance with Intel(R) HPC
                  platform specification. Include Intel(R) HPC Platform
                  Specification meta-packages, Intel(R) Performance
                  Libraries and Intel(R) Distribution for Python
--intel_psxe_runtime Install Intel(R) Parallel Studio XE Runtime 2018
--intel_comp_meta   Install Intel(R) HPC Platform Specification
                  meta-packages
--intel_mpi       Install Intel(R) MPI 2018 (Recommended version
                  for running MPI jobs on Google Cloud)
--intel_python    Install latest Intel(R) Distribution for Python
```

Todas as opões podiam ser necessárias, pelo que foram todas instaladas com o comando seguinte, notando a opção de timeout de 1 hora no comando pssh (-t 3600):

```
[user01@hpc11 ~]$ for i in intel_checker intel_compliance intel_psxe_runtime intel_comp_meta intel_mpi intel_python ; do echo $i ; pssh -t 3600 -h hosts.pssh -- sudo google_install_mpi --${i} ; done
```

De seguida foi realizado um reboot de todo o cluster a partir do controller.

NOTA: notar bem a ordem do ciclo “for” que está propositadamente invertida.

```
[user01@hpc11 ~]$ for i in 4 3 2 1  ; do ssh hpc1$i "sudo  reboot "; done
```

Notar que surgem agora as seguintes mensagens do ficheiro /etc/motd:

```
Enable Intel(R) Cluster Checker with:
source /opt/intel/clck/2019.10/bin/clckvars.sh

Enable Intel(R) MPI Libraries and Intel(R) Parallel Studio XE Runtime with:
source /opt/intel/psxe_runtime/linux/bin/psxevars.sh

Enable Intel(R) Distribution for Python with:
export PATH=/opt/intel/intelpython3/bin/:/sbin:/bin:/usr/sbin:/usr/bin
```

No presente teste, o binário “mpirun” ficou instalado em /opt/intel/psxe_runtime_2018.4.274/linux/mpi/intel64/bin/mpirun

A versão instalada foi:

```
Intel(R) MPI Library for Linux* OS, Version 2018 Update 4 Build 20180823 (id: 18555)

Copyright 2003-2018 Intel Corporation.
```

14. TESTE DO MPI (MPITUNE)

A documentação *Best practices for running tightly coupled HPC applications on Compute Engine* refere:

"To manually specify the algorithms and configuration parameters for MPI Collective communication, we recommend using mpitune"

Foram dadas permissões de escrita ao utilizador Linux user01, no controller, para a pasta /opt/intel/psxe_runtime_2018.4.274/linux/mpi/intel64/etc/, através do comando seguinte:

```
[user01@hpc11 etc]$ sudo chown user01:user01
/opt/intel/psxe_runtime_2018.4.274/linux/mpi/intel64/etc
```

Adaptou-se o exemplo da documentação:

When you use `mpitune`, you tune for each combination of the number of VMs and the number of processes per VM. For example, if you're using Intel MPI 2018 versions, you can tune for 22 VMs and 30 processes per VM by running the following:

```
mpitune -hf HOSTFILE -fl 'shm:tcp' -pr 30:30 -hr 22:22
```

Neste exemplo:

- vms=4 (-hr)
- processes per vm=60 (-pr)

Daí os comandos terem sido os seguintes:

```
[user01@hpc11 hpc]$ source /opt/intel/psxe_runtime/linux/bin/psxevars.sh

[user01@hpc11 hpc]$ mpitune -hf hosts.mpi -fl 'shm:tcp' -pr 60:60 -hr 4:4
```

Na pasta /opt/intel/psxe_runtime_2018.4.274/linux/mpi/intel64/etc/ foram encontrados os seguintes ficheiros novos:

- mpiexec_shm-tcp_nn_4_np_240_ppn_60.conf
- mpiexec.conf
- pasta Benchmarks/

15. TESTE COM APLICAÇÃO HPC - LINPACK BENCHMARK

Para testar o ambiente HPC foi escolhido o software *High-Performance Linpack Benchmark* (HPL) que é uma aplicação pouco dependente do IO e um benchmark HPC amplamente conhecido [9].

[9] Consiltar, por exemplo, a lista https://www.top500.org/

O software foi extraído de https://www.intel.com/content/www/us/en/developer/articles/technical/onemkl-benchmarks-suite.html a penúltima versão de 2021. Optou-se por usar esta versão e não a última pois pareceu ser a versão com data mais próxima da documentação da Google, do que a última versão disponível.

O software é fornecido pré-compilado e foi extraído para pasta /hpc/user01/T1/, que após extração com "gunzip" e "tar" acabou por ficar instalado em:

/hpc/user01/T1/l_onemklbench_p_2021.2.0_109/benchmarks_2021.2.0/linux/mkl/benchmarks/mp_linpack/

Notar que o diretório /hpc está partilhado por NFS no cluster todo e assim o executável do HPL fica acessível em todo o custer.

Para gerar a configuração HPL, para o ficheiro HPL.DAT, usou-se o utilitário online em https://www.advancedclustering.com/act_kb/tune-hpl-dat-file/

Com as seguintes parametrizações:

- Nodes=4
- Cores per node=60
- Memory per node (MB)=200.000 [10]
- O block size (NB) ficou como estava que era 192

O ficheiro HPL.dat gerado foi o seguinte:

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
289536         Ns
1            # of NBs
192           NBs
0            PMAP process mapping (0=Row-,1=Column-major)
1            # of process grids (P x Q)
15            Ps
16            Qs
16.0         threshold
1            # of panel fact
2            PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
4            NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
1            RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
1            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
1            DEPTHs (>=0)
```

[10] Cada servidor tinha 240 GB de RAM.

```
2             SWAP (0=bin-exch,1=long,2=mix)
64            swapping threshold
0             L1 in (0=transposed,1=no-transposed) form
0             U  in (0=transposed,1=no-transposed) form
1             Equilibration (0=no,1=yes)
8             memory alignment in double (> 0)
##### This line (no. 32) is ignored (it serves as a separator). ######
0                               Number of additional problem sizes for
PTRANS
1200 10000 30000                values of N
0                                 number of additional blocking sizes
for PTRANS
40 9 8 13 13 20 16 32 64        values of NB
```

NOTA: sobre o Linpack [11]: LINPACK is a software library for performing numerical linear algebra on digital computers (…) The parallel LINPACK benchmark implementation called HPL (High Performance Linpack) is used to benchmark and rank supercomputers for the TOP500 list.

O comando para correr o HPC (na pasta /hpc/user01/T1/l_onemklbench_p_2021.2.0_109/benchmarks_2021.2.0/linux/mkl/benchmarks/mp_linpack/) foi o seguinte:

```
#export
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/intel/psxe_runtime_2018.4.274/linux/mpi/intel64/lib

mpirun=/opt/intel/psxe_runtime_2018.4.274/linux/mpi/intel64/bin/mpirun


$mpirun -hostfile   hosts.mpi \
 -genv I_MPI_FABRICS "shm:tcp" \
  -np 240  \
   -ppn 60 \
   -tune mpiexec_shm-tcp_nn_4_np_240_ppn_60 \
 ./xhpl_intel64_dynamic
```

NOTA: o mpirun corre mesmo que não encontre o ficheiro de “tune”, embora com um warning. Deve-se tomar a atenção que não surge tal warning.

Explicação dos parâmetros:

- “-hostfile   hosts.mpi \”
    - Ficheiro com a lista de nós no cluster que neste teste teve o conteúdo seguinte:

```
hpc11
hpc12
hpc13
hpc14
```

[11] https://en.wikipedia.org/wiki/LINPACK

- "-genv I_MPI_FABRICS "shm:tcp" \"
    - A documentação *Best practices for running tightly coupled HPC applications on Compute Engine* indica o seguinte:

For Intel MPI 2018 versions, specify the underlying communication fabrics to use TCP over the shared-memory-only provider when running on Google Cloud:

```
mpirun -hostfile HOSTFILE -np NUM_PROCESSES \
    -ppn PROCESSES_PER_NODE -genv I_MPI_FABRICS "shm:tcp" APPLICATION
```

- "-np 240 \" = número de processos que são 4 nós vezes 60 core-CPU
- "-ppn 60 \" = número processos por cada nó que são 60
- "-tune mpiexec_shm-tcp_nn_4_np_240_ppn_60 \" = aproveitamento da configuração do mpitune realizada anteriormente
- " ./xhpl_intel64_dynamic" = binário que corre em todo o cluster
    - NOTA: notar que o binário é partilhado por NFS

Notas adicionais:

NOTA: a variável LD_LIBRARY_PATH precisa ter o local das libs MPI

Durante o teste, que demorou várias horas, um "top" em cada um dos quatros nós apresentou um load de CPU elevado e a informação "61 running", ou seja, que estavam a correr 60 processos HPC mais um processo que era o próprio "top".

```
user01@hpc11:/hpc/user01/T1/l_onemklbench_p_2021.2.0_109/benchmarks_2021.2.0/linux/mkl/...
top - 21:12:19 up  3:52,  2 users,  load average: 776.92, 742.20, 764.52
Tasks: 555 total,  61 running, 494 sleeping,   0 stopped,   0 zombie
%Cpu(s): 30.4 us, 56.5 sy,  0.0 ni, 13.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
KiB Mem : 24745072+total, 59288596 free, 18779332+used,   368800 buff/cache
KiB Swap:        0 total,        0 free,        0 used. 58751808 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 5714 user01    20   0 3964308   3.0g   6052 R 683.1  1.3  27:46.14 xhpl_intel+
 5715 user01    20   0 3964028   3.0g   5976 R 239.5  1.3  25:38.98 xhpl_intel+
 5716 user01    20   0 3964028   3.0g   5944 R 123.6  1.3  23:28.77 xhpl_intel+
 5717 user01    20   0 3964164   3.0g   5956 R 123.6  1.3  23:34.30 xhpl_intel+
 5718 user01    20   0 3935492   2.9g   6088 R 123.6  1.2  23:25.17 xhpl_intel+
 5720 user01    20   0 3935492   2.9g   5948 R 123.6  1.2  22:46.13 xhpl_intel+
```

O resultado da run foi de 1416 G-FLOP, ou seja, 1,4 T-FLOP para os 4 nós. Demorou mais de 3 horas a correr o teste. Output do resultado:

```
================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR11C2R4      289536   192    15    16           11415.41             1.41752e+03
HPL_pdgesv() start time Mon May  2 09:09:46 2022

HPL_pdgesv() end time   Mon May  2 12:20:02 2022

--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=   1.44052599e-03 ..... PASSED
```

```
================================================================================

Finished      1 tests with the following results:
              1 tests completed and passed residual checks,
              0 tests completed and failed residual checks,
              0 tests skipped because of illegal input values.
--------------------------------------------------------------------------------

End of Tests.
```

De acordo com a documentação, "*C2 instances have up to 60 vCPUS (30 physical cores)".*

Assim 1416 G-FLOP são a distribuir por 120 CPU_core (4 VMs x 30), o que prefaz 11,8 G-FLOP por core.

De acordo com a documentação, *"C2 instances … leverage 2nd Gen Intel Xeon Scalable Processors (Cascade Lake)"*.

Algumas referências online referem 12 G-FLOP teóricos por core, pelo que um valor de 11,8 não se afasta muito dos valores esperados.

Em todo o caso, não é objetivo deste teste realizar uma configuração otimizada mas somente realizar uma prova de conceito. Considera-se que 1,4 T-FLOP para os 4 nós de 120 CPU-Core constitui um desempenho com interesse para muitas aplicações de HPC.

Admitindo escalabilidade linear para 11 nós, então, com 11 nós computacionais será possível obter 3,9 T-FLOP.

16. CHECKLIST

A documentação *Best practices for running tightly coupled HPC applications on Compute Engine* apresenta a seguinte checklist que foi verificada:

Item	**Item**	**Verificação / Comentário**
Compute Engine configuration	Use compute-optimized instances	Sim. Foi usado "--machine-type=c2-standard-60"
	Use compact placement policy	Sim. Foi usada uma group-placement especifica
	Disable Hyper-Threading	Sim. Correu-se script com opção "nosmt - Disable simultaneous multi threading"
	Adjust user limits	Sim. Correu-se script com opção "limits - Change the system ulimits"
	Set up SSH host keys	Sim
Storage	Choose an NFS file system or parallel file system	**Não.**

		Por opção, instalou-se um servidor NFS simples no Controller do cluster
	Choose a storage infrastructure	Idem. Ver acima.
Network settings	Increase tcp_*mem settings	Sim. Correu-se script com opção "tcpmem - Increase memory for TCP"
	Use the network-latency profile	Sim. Correu-se script com opção "networklatency - Enable busy polling and low network latency tuned profile"
MPI libraries and user applications	Use Intel MPI	Sim
	Use mpitune to do MPI Collective tuning	Sim
	Use MPI/OpenMP hybrid mode	Presume-se que sim, pois usou-se a configuração recomendada "-genv I_MPI_FABRICS "shm:tcp"" para a versão 2018
	Compile applications using vector instructions and the Math Kernel Library	Não aplicável. Neste teste não foi compilada a aplicação HPC, mas sim usado o HPC da Intel, pré-compilado
	Use appropriate CPU numbering	Não aplicável no teste realizado
Security settings	Disable Linux firewalls	Sim. Correu-se o script com opção "nofirewalld - Disable firewalld"
	Disable SELinux	Sim. Correu-se script com opção "noselinux - Disable SE Linux"
	Turn off Meltdown and Spectre mitigation	Sim. Correu-se script com opção "nomitigation - Disable CPU vulnerabilities mitigations"

Verificações adicionais:

- A documentação recomenda usar o Google Virtual NIC (gVNIC). Para verificar basta correr o comando seguinte: "$ sudo dmesg |grep gvnic|grep eth", que resulta no output do tipo:
  - "… gvnic 0000:00:04.0 eth0: Device link is up."
  - Se não surgir uma mensagem desse tipo então o device não estará configurado.
- NTP. É conveniente também verificar que o relógio de sistema está sincronizado em todo o cluster

17. DESLIGAMENTO E CANCELAMENTO DE CUSTOS

Sempre que os recursos virtuais não estavam a ser usados foi feito o "stop" das VMs para evitar custos de os manter a correr na GCP.

No fim dos testes foi fechada a conta para evitar a hipótese de manter uma conta aberta com eventuais custos futuros.

17.1 Cancelamento do projeto GCP

Instruções sobre o cancelamento (shutdown) de projetos GCP podem ser encontradas em:

https://cloud.google.com/resource-manager/docs/creating-managing-projects

17.2 Fecho de conta de faturação

Instruções sobre de conta de fatura podem ser encontradas em:

https://cloud.google.com/billing/docs/how-to/manage-billing-account

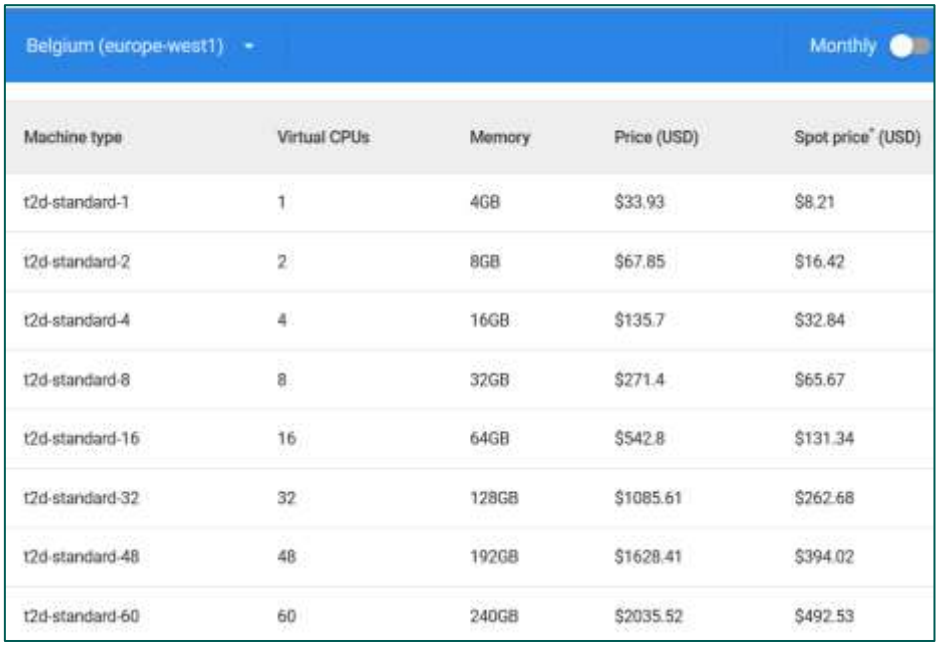
18. CONCLUSÕES

O presente teste seguiu a documentação da Google *Best practices for running tightly coupled HPC applications on Compute Engine* constituindo uma prova de conceito simplificada de configuração HPC na cloud pública GCP. Foi configurado um cluster de 4 nós com 120 CPU- Cores do tipo "Intel(R) Xeon(R) CPU @ 3.10GHz" com aproximadamente 1 TeraByte de RAM. A documentação refere que é possível lançar simplificadamente clusters deste tipo até 11 nós, o que corresponderia a um cluster de 330 CPU-Cores.

Para esta configuração foi necessário obter aprovação de aumento de quotas da Google para várias configurações, uma delas com intervenção da área comercial [12]. Essa aprovação foi dada em 2 dias úteis.

Os preços podem ser consultados online ou, preferencialmente, através dos contactos comerciais da Google. A tabela seguinte, pública, é disso uma aproximação, sendo relevante o "machine type" "-60" com o preço de 2.035 USD por mês. Um cluster de 11 nós teria, portanto, um custo mensal de 22.385 USD, ao qual deve ser adicionado o custo do storage e funções acessórias, como login-nodes, compile-nodes, base dados, LDAP, etc.

[12] "*before we can proceed with our evaluation, your requested quota increase for your project … will require your Sales team's support*"



Belgium (europe-west1) Monthly

Machine type	Virtual CPUs	Memory	Price (USD)	Spot price* (USD)
t2d-standard-1	1	4GB	$33.93	$8.21
t2d-standard-2	2	8GB	$67.85	$16.42
t2d-standard-4	4	16GB	$135.7	$32.84
t2d-standard-8	8	32GB	$271.4	$65.67
t2d-standard-16	16	64GB	$542.8	$131.34
t2d-standard-32	32	128GB	$1085.61	$262.68
t2d-standard-48	48	192GB	$1628.41	$394.02
t2d-standard-60	60	240GB	$2035.52	$492.53

Nem sempre foi possível lançar o *cluster* devido ao erro "*The zone 'projects/…zones/europe-west…' does not have enough resources available to fulfill the request*". Em alguns casos as VMs não podiam ser lançadas. Noutro caso já estavam criadas, mas não foi possível fazer o "start" delas. Essas limitações foram temporárias.

Correu-se um teste Linpack - *High-Performance Linpack Benchmark (HPL)* – que apresentou um resultado de 1,4 T-FLOP para 4 nós de 120 CPU_Core do tipo *2nd Gen Intel Xeon Scalable Processors (Cascade Lake)* a 3,1 GHZ. Admitindo escalabilidade linear de 4 para 11 nós, então, com 11 nós computacionais será possível obter 3,9 T-FLOP. Não se pretendeu com este teste realizar uma instalação otimizada pelo que poderia ser possível obter maior performance, analisando alternativas de configuração e compilando o HPL em vez de usar uma versão pré-compilada.

O resultado do presente teste não constitui uma instalação de produção, faltando desenvolvimentos importantes como:

- Um sistema de storage de alto desempenho, que em muitas aplicações HPC é absolutamente necessário e cuja oferta existirá na GCP.
- Uma gestão de utilizadores completa, para controlo de acessos, contabilização de utilização e outras funções.
- Um sistema de gestão de filas de trabalho, tipo Slurm ou outro, bem como um sistema compilação de versionamento de software e *libs*.
- A aplicação de uma política de segurança com controlo mais fino dos sistemas expostos à Internet.

O teste foi concluído com sucesso. A utilização de HPC em ambiente cloud não é, de resto, um tema inovador sendo tratado há vários anos em conferências de HPC [13]. Poderá haver aplicações que precisem de performance mais apurada que exiga uma instalação com *hardware* controlado diretamente pelo centro operacional. Os custos podem ser facilmente calculados através da projeção dos tarifários cloud publicados online. O planeamento da capacidade disponível na zona da cloud pretendida, deve ser uma preocupação a ter, pois, como se observou neste teste, pode haver limitações pontuais de capacidade nas zonas da cloud.

As competências técnicas necessárias para instalar um cluster HPC em cloud são principalmente de sysadmin Linux e são, essencialmente, as mesmas de uma instalação física normal, substituindo a complexidade da gestão física de cablagens, gestão de calor, gestão de consolas IPMI, gestão de redes e outras funções, pela gestão das primitivas de configuração de recursos cloud. Um sysadmin Linux que já tenha instalado um cluster MPI conseguiria seguir o documento Google *Best practices for running tightly coupled HPC applications on Compute Engine* e criar um ambiente computacional em alguns dias. Esta tarefa poderá não estar acessível aos conhecimentos típicos de um investigador não-informático, que certamente precisaria da assistência de serviços de informática especializados. A configuração de um cluster MPI em cloud, em vez de ser uma instalação física, pode ter uma vantagem importante no aspeto de reutilização e replicação de configurações, pois será possível criar imagens de ambientes completos HPC, pré-configurados com as parametrizações da organização, com a possibilidade de se fazer um *deploy* rápido para um determinado fim específico.

A documentação refere que é possível lançar simplificadamente até 11 nós de 30 core-CPU cada nó [14] do tipo Intel Scalable Processor (Cascade Lake), o que prefaz um cluster de 330 core-CPU, o que, para algumas aplicações HPC é insuficiente, mesmo que seja aceitável a arquitetura de hardware disponível em cloud.

19. ACRÓNIMOS

Lista de acrónimos e definições:

- Controller = Serviço “mestre” de configurações do cluster, neste exemplo foi o HPC11
- GCP=Google Cloud Platform
- HPC=High Performance Computing
- SO = Sistema Operativo (Linux, Windows, etc.)
- VM = Virtual Machine
- Tera (Adaptado de https://pt.wikipedia.org/wiki/Tera). 1 T = 10^12

[13] p. ex. Supercomputing – SC* - *SuperCompCloud: Workshop on Interoperability of Supercomputing and Cloud Technologies* - https://sciencenode.org/feature/SuperCompCloud%20Workshop%20at%20SC19.php

[14] Corresponde a 60 v-CPU por nó com hyper threading ativado

Prefixo		10^n
Nome	Símbolo	
iota	Y	10^{24}
zeta	Z	10^{21}
exa	E	10^{18}
peta	P	10^{15}
tera	T	10^{12}
giga	G	10^{9}
mega	M	10^{6}
quilo	k	10^{3}
hecto	h	10^{2}
deca	da	10^{1}

- T-FLOP – Tera FLOP - FLoating-point Operations Per Second
    - G-FLOP – Giga FLOP

20. CONTROLO DE VERSÕES

Para correções ou melhorias desde documento contactar RNCA@fccn.pt

Versão	Alterações	Autores, revisores
V1a de 3 de Maio de 2022	-	Unidade de Computação Científica Nacional (FCCN) da FCT (João Pagaime) Revisão da estrutura do texto: Catarina Guerreiro (FCT-FCCN)