FALL : Du VDI au HPC, mutualisation d'une infrastructure GPU

Guenael Sanchez
Université Grenoble Alpes – DGDSI
43,rue des Mathématiques
38 400 Saint-Martin d'Hères

Bruno Bzeznik
Université Grenoble Alpes– GRICAD
Bâtiment IMAG – 700 Avenue centrale
38 400 Saint-Martin d'Hères

Résumé

Il ne manquait plus qu'une quatrième saison à l'Université Grenoble Alpes pour terminer sa pizza Vivaldi. SUMMER & WINTER pour le stockage & le cloud mutualisé, SPRING pour le réseau DataCentre, FALL boucle enfin la série avec une plateforme VDI survitaminée et mutualisée.

Avec une douzaine de nœuds riches en GPU et rythmés par VMware Horizon, FALL offre depuis la rentrée 2020 un service de virtualisation d'applications et de postes de travail aux différentes entités de l'établissement.

Une plateforme VDI, ça coûte une blinde peuchère ! Ça vaut vraiment le coup ? Le retour sur investissement est complexe à déterminer !

L'usage du VDI étant concentré principalement sur les horaires de bureau, que faire de tous ces temps "morts" de GPU ? Miner du Bitcoin ? Et pourquoi pas du calcul…

Le mésocentre de calcul intensif grenoblois n'en est d'ailleurs pas à sa première expérience de ce genre. Dès les années 2000, le projet « CIMENT » mettait en place sa grille de calcul « CiGri » basée sur le principe de la récupération des cycles inutilisés. (rien à voir avec les vélos)

Pour l'utilisateur calcul, FALL est une grappe de calcul dans laquelle les ressources GPU sont volatiles. Bien entendu, cela ne s'adresse qu'à une petite part des utilisateurs, car ce mode de fonctionnement ne convient qu'à certains jobs particuliers.

Alors qu'avons-nous gagné à mutualiser ces ressources GPU ? Comment ça marche techniquement ? Comment tout cela est ressenti côté utilisateurs ? Et du coup, c'est rentable ou pas cette affaire ?

Mots-clefs

VDI, HPC, GPU, VMware, Horizon, Nvidia

1 Contexte grenoblois

En 2016, les trois Universités grenobloises ont fusionné pour donner naissance à l'Université Grenoble Alpes (UGA). Un an auparavant, les différentes DSI planchaient sur le rapprochement des différentes infrastructures qui n'étaient pas encore mutualisées, ainsi que sur les choix concernant les futures plateformes de l'UGA.

1.1 Un peu d'historique

Selon les thématiques, la mutualisation était très variable, très peu développée pour la virtualisation de serveurs par exemple, ou ancrée depuis plusieurs années pour le mésocentre de calcul.

1.2 Les services de site

Sous l'égide de l'UAR (Unité d'Appui à la Recherche) GRICAD (**G**renoble Alpes **R**echerche – **I**nfrastructure de **CA**lcul intensif et de **D**onnées) l'UGA propose un catalogue de services à destination de tous les utilisateurs de l'Université. Le fil rouge de ces services est la mutualisation des besoins ainsi que la mutualisation des équipes techniques opérant ces services sous la forme de comités techniques.

1.2.1 CIMENT et GRICAD : Le mésocentre

CIMENT (**C**alcul **I**ntensif **M**odélisation **E**xpérimentation **N**umérique et **T**echnologique) est le mésocentre de calcul de l'université de Grenoble créé dans les années 2000. Aujourd'hui, pleinement intégré à GRICAD, CIMENT est l'acronyme associé à toutes les plateformes de calcul intensif et systèmes de stockage associés accessibles par les chercheurs du bassin Grenoblois.

1.2.2 SPRING : Le réseau des Datacentre

La plateforme SPRING a pour but d'imaginer, de déployer et d'opérer, grâce à un comité technique, une infrastructure SDN (Software Defined Network) mutualisée pour les Datacentres. SPRING fournit un haut niveau de service et de performance aux différents services de site. SPRING a été présenté aux JRES 2017 [1]

1.2.3 SUMMER : Le stockage des données

SUMMER propose des solutions de stockage et de sauvegarde réparties dans trois centres de données de l'UGA. Avec presque 10 Po de capacité utile, SUMMER fournit de la volumétrie pour la recherche, mais aussi pour la pédagogie et les services administratifs. La solution a également été présentée aux JRES 2017 [2]

1.2.4 WINTER : Le cloud privé

WINTER héberge les besoins de machines virtuelles. Avec 1000 VMs en fonctionnement, l'UGA permet à ses utilisateurs de bénéficier d'un cloud privé et sécurisé. WINTER a également été présenté aux JRES 2017 [3]

2 La plateforme FALL

L'année universitaire 2019/2020 a remis en haut de la liste les discussions concernant les solutions de virtualisation des postes de travail (VDI).

2.1 Genèse du projet

L'Institut Universitaire de Géographie Alpine (IUGA), ainsi que les IUT avaient besoin de faire fonctionner des applications pédagogiques nécessitant des ressources graphiques (GPU) importantes. S'est posée la question d'équiper les salles pédagogiques de machines performantes,

ou d'étudier des solutions VDI. La DSI de l'UGA, en s'inspirant des services de site existants, a porté la création du projet FALL afin de mutualiser les besoins VDI, opéré là aussi par un comité technique incluant les différents acteurs concernés.

La pandémie de Covid-19 est arrivée et le besoin de pouvoir accéder à ces ressources pédagogiques à distance s'est considérablement renforcé.

2.2 Tour d'Horizon

Fort des compétences déjà acquises sur les solutions VMware utilisées sur la plateforme WINTER, le comité technique s'est rapidement orienté vers la solution de VDI Horizon éditée par VMware.

Après plusieurs échanges positifs et constructifs avec d'autres Universités comme Aix-Marseille ou Paris Nanterre, une solution a été retenue. 10 Serveurs DELL, chacun équipé de 48 cœurs à 3GHz, de 768 Go de mémoire vive, et de plusieurs dizaines de To de stockage SSD haute performance. Pour le côté ressources graphiques, notre choix s'est porté sur des cartes T4 du constructeur Nvidia. Chaque serveur est équipé de 4 cartes de 16 Go.

Côté logiciel VMware, nous sommes partis sur 400 licences Horizon Entreprise et 200 licences Nvidia GRID. Ces licences s'entendent comme un maximum d'utilisations en simultané.

La licence Entreprise inclut tous les logiciels de virtualisation sous-jacents nécessaires au fonctionnement d'Horizon (vSphere, vSAN, vCenter). Elle inclut également des composants utiles à la virtualisation d'applications (AppVolume) et la personnalisation de l'environnement utilisateur (DEM).

Enfin, les bureaux virtualisés fonctionnant soit sous Linux, soit sous Microsoft Windows, il était nécessaire de s'acquitter d'un certain nombre de licences. Windows Server édition DataCenter + Software Assurance pour les serveurs physiques, permettant ainsi le fonctionnement de machines virtuelles sous Windows. Mais également des licences VDA permettant l'accès à des bureaux virtuels, ainsi que des droits d'utilisations (CAL) pour utiliser des services comme ActiveDirectory.

Dans le cadre de l'UGA, nous avons fait le choix d'intégrer tous ces coûts de licences Microsoft dans notre contrat EES. Ce contrat intégrait précédemment d'autres logiciels comme Office 365, Windows Education, ou du support Software Assurance, etc. L'accumulation des licences au sein du même contrat a permis de maximiser les taux de remise.

2.3 Complexité du calcul de ROI

Le coût initial d'une plateforme VDI est important. Les compétences techniques nécessaires à son installation et son fonctionnement opérationnel le sont également. Pour une petite entité, ce ticket d'entrée est trop important, mais dans le cadre d'une mutualisation des besoins, des investissements et des ressources humaines, il est intéressant d'aborder ce calcul complexe de retour sur investissement, d'économies d'échelle potentielles, etc.

Nous avons calqué notre réflexion sur le fonctionnement financier des autres services de sites de l'UGA. L'établissement, au travers de la DSI supporte les investissements, charge ensuite au comité technique de trouver un modèle économique viable de refacturation du service apporté aux différentes entités.

L'équation doit permettre de proposer un prix attractif pour favoriser l'adhésion et la mutualisation, mais aussi générer un certain équilibre financier global sur le long terme (investissements réalisés pour 5 à 7 années d'exploitation).

Le modèle doit permettre également l'implication opérationnelle des équipes locales, la mutualisation devant permettre, à terme, un allègement de la charge des équipes, tout en cultivant un intérêt technique. Le but n'est pas de déshabiller techniquement les équipes de proximité, au profit d'une centralisation des compétences au sein de la DSI. Le VDI comporte d'ailleurs un lien fort avec les équipes pédagogiques locales, quant à l'usage des logiciels pédagogiques.

2.4 Spécificités des GPU

Les ressources GPU ont quelques spécificités. Premièrement, seule une partie des utilisations de la plateforme nécessitent l'utilisation de ces ressources. Inutile donc de provisionner de la ressource GPU pour tous les utilisateurs se connectant à la plateforme.

Il existe plusieurs typologies d'accès à des ressources VDI. La figure ci-dessous montre le fonctionnement que nous avons retenu à l'heure où nous écrivons ces lignes.

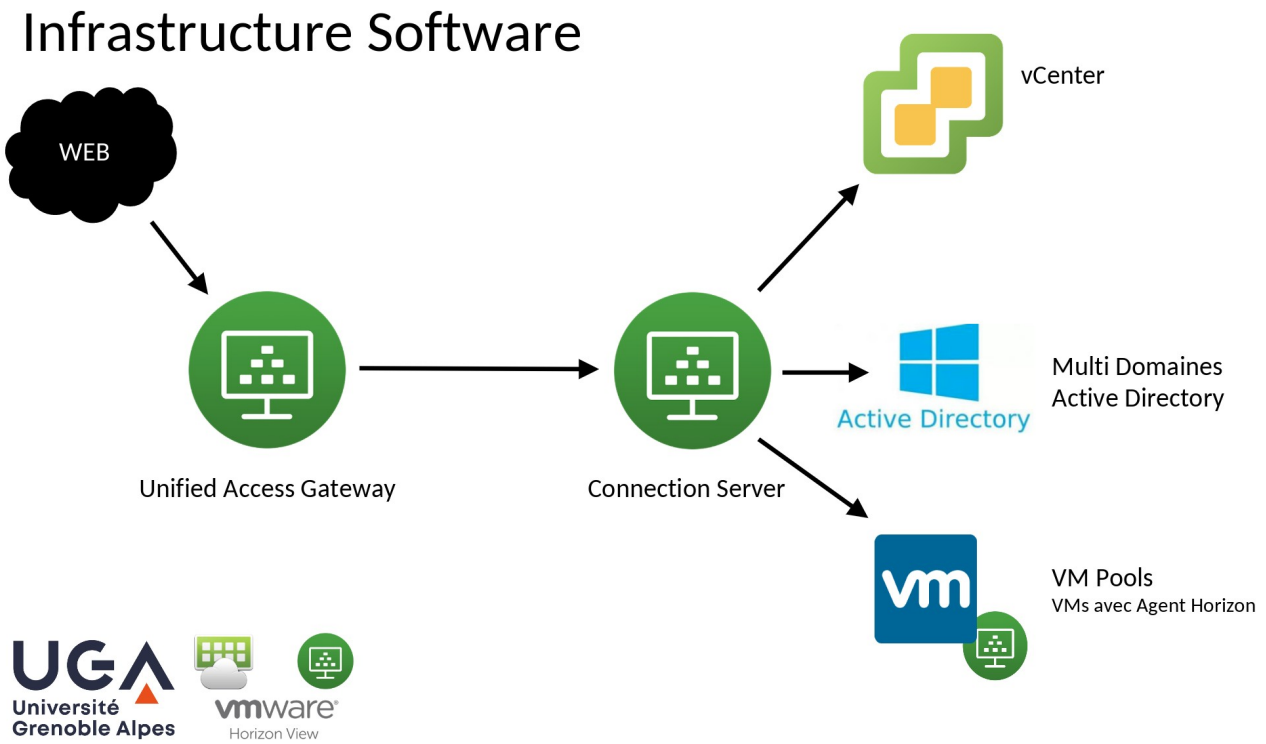

Figure 1: Topologie de l'infrastructure logicielle FALL

L'utilisateur se connecte, au travers d'un portail web, et se retrouve à piloter une machine virtuelle de l'infrastructure FALL. Au moment de la connexion, l'utilisateur peut choisir entre différents bureaux, en fonction de ses droits, et de ses besoins applicatifs.

Dans le cas où l'utilisateur souhaite utiliser une application graphique nécessitant l'utilisation de ressources GPU, ce dernier est renvoyé vers un pool de VM déjà configurées aux caractéristiques particulières.

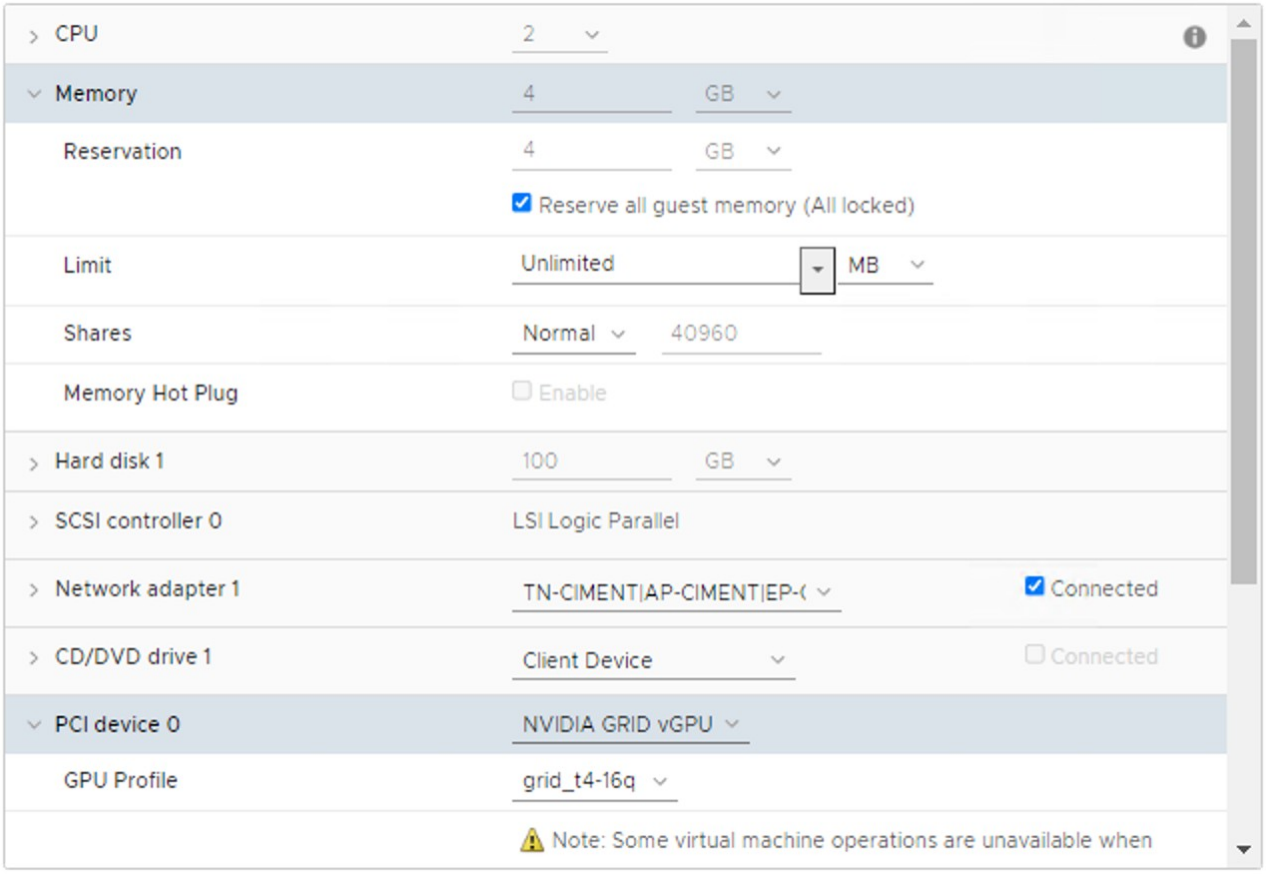

Figure 2: Caractéristiques des machines virtuelles avec GPU

On peut noter deux points :

1. La mémoire vive virtuelle est réservée pleinement. Si la VM est prévue avec 8 Go de RAM, alors 8 Go de RAM sont réservés sur le serveur. Contrairement aux autres VMs dites « classiques » où seul l'espace mémoire réellement utilisé par le système d'exploitation client est réservé sur le serveur. Cette spécificité est une conséquence du point 2.
2. La ressource GPU apparaît comme un périphérique PCI virtuel, auquel on va joindre un « profil mémoire graphique ».

– Du point de vue du système d'exploitation client, on dispose d'un périphérique PCI graphique, qui sera exploité par le driver Nvidia GRID. Quand le pilote démarre, il négocie un jeton de licence d'utilisation auprès d'un serveur, et réserve lui aussi pleinement une quantité de mémoire GPU.

– Du point de vue du serveur de virtualisation, la VM va envoyer des commandes sur un bus PCI virtuel. Ces commandes seront interprétées par le driver Nvidia GRID installé dans vSphere, avant d'être renvoyées vers la carte physiquement installée dans le serveur.

Ce modèle de virtualisation des ressources graphiques existe parmi d'autres solutions qui sont abordées au chapitre 4.1 Choix techniques.

On peut retenir qu'une VM avec ressources GPU monopolise de la mémoire vive du serveur, de la mémoire GPU, ainsi qu'un jeton de licence vGPU, et ce, que la VM soit utilisée ou non !

3 Le mésocentre CIMENT

Le mésocentre de calcul intensif CIMENT dispose en permanence de plusieurs machines de calcul en exploitation. Environ 6000 cœurs de calcul de type Xeon, 50 GPUs de type V100 ou A100, et 10Po de stockage (du scratch ultrarapide distribué sur NVMe, à l'archivage sur disques) sont utilisés de manière intensive par de nombreuses simulations numériques ou analyses de données lancées par une grande communauté multi-disciplinaire d'environ 600 utilisateurs actifs. La charge d'un tel centre de calcul est très variable en fonction du temps et de la plate-forme utilisée.

Aussi, dès les années 2000, CIMENT s'est doté d'outils permettant une gestion des ressources efficace, dont une grille de calcul, CiGri, permettant de répartir les jobs de type multi-paramétriques sur les petites partitions laissées libres entre les gros jobs de calcul parallèles. Par le passé, nous avons déjà mis en place des configurations permettant de transformer des salles de PC destinées à la formation en grappe de calcul virtuelle nocturne [5]. Cette utilisation opportuniste des ressources de calcul disponibles « ailleurs » permet d'augmenter provisoirement la capacité de production du centre de calcul, et permet aussi souvent donner la priorité aux « gros » jobs parallèles sur les plateformes dédiées et d'éviter que ces dernières ne soient trop encombrées de jobs de type « embarrassingly-parallel ».

3.1 Ordonnancement et Planification

Pour la gestion des taches (appelés aussi « jobs ») et des ressources de calcul, CIMENT utilise le gestionnaire OAR [1]. Chaque cœur de CPU correspond à une ressource qui est disponible ou non, en fonction de son état (fonctionnement normal, en cours de maintenance, en mode économie d'énergie…) et de son occupation par des jobs de calcul. Les ressources sont fournies par des serveurs physiques appelés « nœuds », généralement interconnectés par un réseau rapide à faible latence (par exemple Infiniband ou OmniPath).

Un job est en général un script écrit par l'utilisateur pour lancer automatiquement un programme de calcul. Chaque job a des caractéristiques définies à sa soumission : le nombre de ressources nécessaires, sa durée maximum (le « walltime ») et des contraintes sur les « propriétés » des ressources demandées. Les jobs sont d'abord placés dans une file d'attente puis ordonnancés en fonction de priorités dépendant de l'utilisateur (« fairsharing » en général) et des caractéristiques du job. Dès que les ressources nécessaires sont disponibles, les jobs sont lancés. Ainsi, en cas de congestion, une partie des jobs se trouve dans l'état « Waiting » et ils démarreront automatiquement au plus tôt.

Pour qu'un job de calcul puisse fonctionner, il faut qu'il puisse accéder à des fichiers d'entrée et qu'il puisse écrire sur des systèmes de stockage temporaire performants (appelés « scratch ») et

1 OAR : http://oar.imag.fr

enfin qu'il puisse écrire les fichiers de sorties. Ainsi, chaque nœud de calcul correspondant à une grappe est généralement connecté à un ou plusieurs systèmes de fichiers distribués.

Dans CIMENT, on compte au moins 4 types de systèmes de fichiers accessibles depuis chaque nœud de calcul : `/home` généralement en NFS, un scratch local au nœud de calcul (typiquement `/tmp`), un scratch distribué sur tous les nœuds de la grappe (BeeGFS[2] ou Lustre[3]) et enfin un stockage de type cloud largement accessible par les différentes grappes (géré par iRods[4]).

Enfin, un ou plusieurs répertoires contiennent les applicatifs nécessaires à l'exécution des jobs. Nous utilisons principalement Nix, Guix et des installations ad hoc ; tout ceci est partagé en NFS sur chaque nœud de calcul.

3.2 Spécificités des calculs GPU

Les nœuds de calcul équipés de GPU sont similaires à n'importe quel autre nœud de calcul, étant donné que les GPU ne sont pas autonomes et nécessitent d'être pilotés par un processeur central et que les données doivent transiter via la mémoire centrale. Aussi les nœuds dits « GPU » sont vus par OAR comme des nœuds ayant des propriétés particulières et contenant un certain nombre de « gpudevice ». Chaque gpudevice est associé à un certain nombre de cœurs de CPU et un volume de mémoire système. Ainsi, par exemple, un job nécessitant 2 GPU sur un même nœud se verra également octroyer 8 cœurs et 48Go de RAM système. OAR permet l'isolation dans des cgroup[5] et ne rendra visibles que les gpudevice associés au job courant.

3.3 Intégration des ressources de FALL

CIMENT fourni plusieurs types de GPU, actuellement des **A100** et des **V100**. Ce sont des cartes très performantes, mais également très onéreuses et énergivores. FALL dispose actuellement de petites cartes **T4** plus modestes en termes de performances, mais également moins énergivores (75 W chacune) et disponibles hors des périodes de formation.

Certains codes de calcul sont aussi performants sur T4 que sur V100, il s'agit notamment des codes d'inférence en intelligence artificielle. D'autres codes affichent des performances diminuées de moitié. Mais certains de ces jobs nécessitent de tourner de nombreuses fois, sur un ensemble de paramètres par exemple. Même si ces jobs de calcul tournent moins vite sur ces cartes, leur nombre est important et il paraît intéressant de viser à les exploiter pendant leurs périodes de disponibilité.

2 https://www.beegfs.io/

3 https://www.lustre.org/

4 https://irods.org/

5 https://man7.org/linux/man-pages/man7/cgroups.7.html

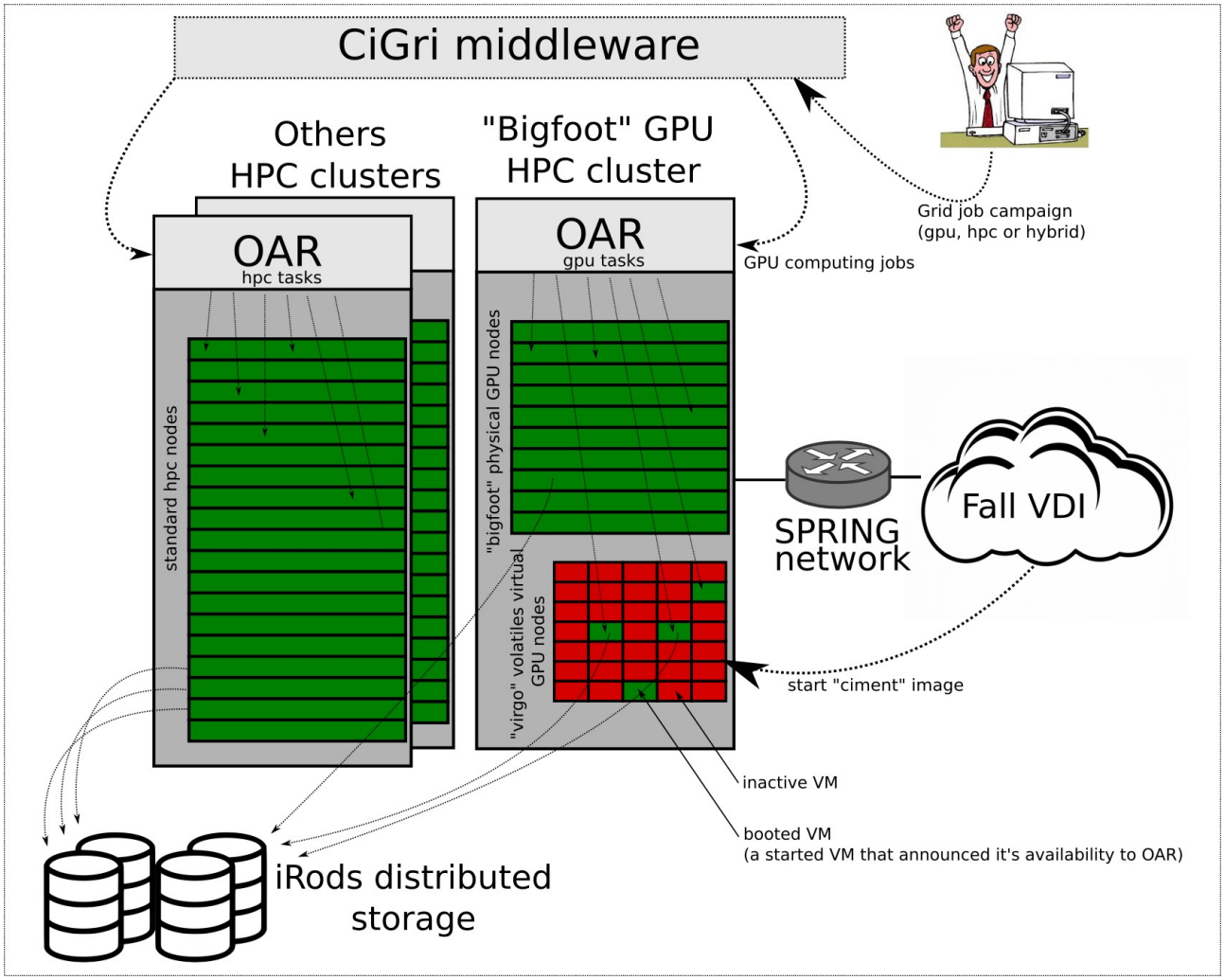

Figure 3: Schéma global Fall VDI → Bigfoot GPU cluster

Du point de vue de CIMENT, il nous a été très facile de configurer une image Ubuntu afin d'en faire un nœud de calcul géré par un gestionnaire OAR. La mise au point de l'image gold a essentiellement consisté en l'installation du client OAR, la configuration des différents espaces de stockage distribués et la configuration du démarrage et de l'arrêt.

Dans notre cas précis, ce dernier point est la clé du fonctionnement : au démarrage, le nœud de calcul s'annonce à la frontale, qui va déclencher une série de tests et mettre en production les ressources. Les jobs qui étaient en attente et dont les contraintes correspondent à des nœuds de type « FALL » vont pouvoir démarrer automatiquement. À l'arrêt du nœud, la frontale est également notifiée, ce qui va déclencher l'arrêt des éventuels jobs en cours et la mise en indisponibilité des ressources.

Cependant, pour éviter que des jobs ne soient encore en cours au moment de l'arrêt, nous avons la possibilité d'indiquer à OAR jusqu'à quelle heure la ressource sera disponible, ce qui lui permet de ne pas envoyer de jobs qui ne pourraient pas être terminés à temps (exploitation de la propriété

OAR « available_upto »). Un mode de fonctionnement spécifique « best-effort » existe, permettant à l'utilisateur d'indiquer que son job supporte des arrêts intempestifs.

De tels jobs sont très intéressants pour exploiter des ressources volatiles comme les VM de FALL. En général, ces jobs peuvent soit reprendre leur calcul où il en était au redémarrage (*checkpointing* applicatif) ou alors ils sont assez courts et nombreux, ce qui limite la perte de résultats lorsque l'un d'eux est interrompu.

3.4 Outil CiGri

CiGri[6] est un intergiciel de grille de calcul spécialisé dans la gestion de campagnes de jobs multi-paramétriques. Il fait partie de l'écosystème de OAR et permet de soumettre un très grand nombre d'instances du même job sur un ensemble de clusters. Il facilite donc la gestion de ce type de jobs et offre en particulier des fonctionnalités intéressantes pour gérer des jobs best-effort qui peuvent alors remplir toutes les opportunités possibles.

CiGri va notamment gérer de manière automatique la « re-soumission » nécessaire lorsque les jobs ont été prématurément tués par disparition d'une ressource de calcul volatile. Grâce à CiGri, les ressources peuvent être pleinement exploitées, jusqu'à 100 % du temps, le plus difficile étant d'identifier et d'adapter les jobs de calcul qui conviennent à ce mode de fonctionnement. Dans le cas des ressources FALL, CiGri est donc un allié supplémentaire vers l'optimisation de cet usage.

3.5 *Cas d'usage* et considérations énergétiques

Nous nous sommes penchés sur le cas particulier d'une application utilisée en chimie théorique Amber18[7]. Dans notre cas d'usage, cette application affiche ses performances en temps *elapsed* par pas de temps simulé. Nous avons comparé son exécution sur une GPU V100 d'un serveur physique de CIMENT à son exécution sur une GPU T4 d'une VM de FALL :

GPU model	Elapsed/step ms	GPU Mean Power (as reported by nvidia-smi)	Power consumption (Wh) / step
V100 (ciment)	0.33	155 W	1.4e-5
T4 (Fall)	0.66	70 W	1.2e-5

Ce que l'on constate facilement, c'est que si le temps d'exécution est 2 fois plus long sur une T4, la consommation énergétique associée est légèrement plus faible. Bien entendu, cette vérification devrait être faite pour chaque application et il serait intéressant de prendre en compte la consommation totale des serveurs, et pas seulement celle de la carte GPU. Néanmoins, on peut supposer que dans le cas de cette application, il n'y a pas de surcoût énergétique à réaliser les calculs sur FALL.

6 http://ciment.univ-grenoble-alpes.fr/cigri/

7 https://ambermd.org/GPUSupport.php

4 La mutualisation en pratique

L'utilisation du VDI impliquant un certain nombre de temps « morts » où l'infrastructure ne fait quasiment rien, nous nous sommes posé la question de l'opportunité de « rentabiliser » ces périodes (nuits, week-ends, fermetures pédagogiques). L'idée était de mettre à disposition ces ressources du mésocentre CIMENT.

Cependant, dès les premiers mois d'utilisation, nous avons constaté que les étudiants avaient tendance à se reconnecter à la plateforme en dehors des heures de cours. La possibilité d'accéder aux différents logiciels pédagogiques dans un environnement performant, pour continuer à travailler ou pour réviser a vite convaincu les élèves de l'intérêt de la solution. Un ordinateur même basique leur permet dorénavant d'accéder à des applications avancées qui nécessiteraient un ordinateur évolué et onéreux.

4.1 Choix techniques

Compte tenu des éléments abordés au chapitre 2.4 - Spécificités des GPU, il fallait choisir entre plusieurs modes de coopération/cohabitation CIMENT/FALL.

4.1.1 100 % calcul la nuit

Dans ce mode, les serveurs de l'infrastructure FALL s'arrêtent proprement le soir et redémarrent, par exemple via le réseau, sur un système d'exploitation dédié au calcul.

Avantages :

– 100 % des ressources graphiques disponibles pour le calcul sans couche de virtualisation intermédiaire ;

– l'environnement Horizon n'est pas « pollué » par une utilisation tierce atypique.

Inconvénients :

– l'utilisation VDI n'est plus possible la nuit, au détriment d'un usage pédagogique, certes résiduel, mais présent presque tous les jours en petit nombre ;

– le cluster vSAN assurant le stockage hyper-convergé des machines virtuelles n'est pas prévu pour un arrêt complet régulier, une procédure existe, mais elle est peu adaptée à un usage quotidien.

Avec ces deux inconvénients, cette solution n'est pas viable, les serveurs devront continuer à fonctionner sous vSphere pendant la nuit.

4.1.2 vSphere avec GPU en accès direct

vSphere propose deux manières de virtualiser les périphériques PCI présents sur le serveur afin de les présenter aux machines virtuelles.

Le premier mode, dit « *pass-through* » renvoie directement les instructions PCI entre le système d'exploitation de la machine virtuelle et la carte physiquement installée dans le serveur.

Avantages :

– la carte GPU est 100 % utilisable par le système d'exploitation virtualisé. Pas de pertes de performance liées à la virtualisation ou alors de manière négligeable.

Inconvénients :

– le changement de mode entre « *pass-through* » et le mode partagé, abordé au point 4.1.3 - vSphere avec GPU en mode partagé vGPU, nécessite un redémarrage du serveur ;

– une carte de 16 Go de RAM ne peut pas être partagée entre plusieurs VMs.

C'est ce deuxième inconvénient qui, dans notre mode de fonctionnement, élimine cette solution. En effet, beaucoup d'utilisations impliquant de la ressource GPU nécessitent uniquement 2 à 4 Go de mémoire GPU et ne sauraient quoi faire des 16 Go de mémoire disponible par carte. On va donc chercher à concentrer plusieurs utilisations sur la même carte.

La nécessité de redémarrer le serveur pour basculer entre les modes « pass-through » et partagé ajoute beaucoup de lourdeur à une utilisation dans ce mode.

4.1.3 vSphere avec GPU en mode partagé vGPU

Dernier mode envisagé, les ressources GPU sont configurées en mode partagé. Il existe, dans ce mode, 2 fonctionnements disponibles, appelés vSGA (pour *virtual Shared Graphics Acceleration*) et vGPU (pour *virtual* GPU).

Dans le premier cas, c'est un pilote VMware qui est utilisé au sein de la machine virtuelle, dans le second c'est un pilote Nvidia.

Nous simplifions largement ici le débat entre ces deux technologies pour conclure que le mode vSGA est adapté à un usage bureautique « enrichi » (lecture de vidéos, applications type Office sachant tirer parti d'une ressource GPU) tandis que le mode vGPU est adapté à un usage 3D intensif de type modélisation, voire calcul de type « *machine-learning* ».

C'est ce deuxième mode vGPU que nous avons retenu dans notre utilisation. Pour plus de détails, nous vous renvoyons aux différents articles du Blog VMware à ce propos. [4]

Avantages :

– une carte GPU de 16 Go peut être partagée entre 1,2,4,8 ou 16 machines virtuelles, en fonction du profil mémoire attribuée à la VM ;

– on bascule d'un usage VDI classique à un usage CIMENT sans opération particulière côté serveur.

Inconvénients :

– Il faut installer un pilote spécifique Nvidia au sein de la VM. Ce pilote déclenche la réservation d'un jeton de licence flottant pendant la durée de vie de la VM, que l'utilisateur soit connecté ou non !

– Il faut installer un pilote spécifique Nvidia au sein de l'hyperviseur. Le vMotion (migration à chaud) de VMs utilisant des ressources GPU n'est possible qu'entre serveurs utilisant la même version de pilote Nvidia. (À prendre en compte pour les phases de mise à jour)

– On ne peut pas mixer des profils mémoires GPU différents sur une même carte. (exemple : il est impossible d'avoir sur une même carte 1 VM avec une réservation de 8 Go et 2 VMs avec une réservation de 4 Go chacune)

- Une perte de performances GPU liées à l'étage intermédiaire de virtualisation.

Malgré ces inconvénients, c'est ce mode qui répond le mieux à notre besoin de départ pour assurer une cohabitation entre les usages VDI et calcul.

Ce mode nous permet également d'avoir un impact limité sur les configurations serveur tout en limitant les pertes de performances GPU liées à la virtualisation.

4.2 Stratégie globale

Nous avons souhaité tirer parti des mécanismes et des API offertes par Horizon pour faciliter un maximum ce travail de collaboration. Le mécanisme InstantClone d'Horizon repose sur les étapes suivantes :

1. Installation & Personnalisation d'un OS sur une VM qui servira de modèle ;
2. Installation d'un agent Horizon dans l'OS qui servira de relais de communication entre le serveur Horizon et la VM finale ;
3. Installation d'un agent d'annuaire dans l'OS qui permet l'enrôlement de la VM dans un annuaire ActiveDirectory (Agent non utilisé au final, mais fluidifie le processus) ;
4. Création d'un snapshot à froid une fois un état satisfaisant atteint.

Dans Horizon, on sélectionne ensuite le snapshot en question, le nombre de machines souhaitées, et c'est lui qui pilotera le clonage/individualisation des machines virtuelles sur la base du snapshot. Les machines sont donc jetables, et sont recréées systématiquement sur la même base.

Par la suite, si l'on souhaite rafraîchir l'image de base :

1. Mise à jour des logiciels dans la VM modèle ;
2. Nouveau snapshot à froid.

Dans Horizon, on indique le nouveau snapshot de référence, les nouvelles VMs clonées seront alors issues de ce dernier instantané.

4.2.1 Image gold HPC

La construction de VM modèles dans Horizon obéit à un certain nombre de prérequis bien documentés. Au moment de la rédaction de cet article, nous utilisons la dernière version disponible d'Horizon 2111 (8.3).

La matrice de compatibilité avec les OS Linux nous oriente rapidement vers la distribution Ubuntu. La matrice de compatibilité du driver Nvidia ne nous laisse que la version 20.04 LTS comme choix compatible.

L’installation de l’OS de base est somme toute assez commune. L’installation du pilote Nvidia nécessite de placer le *driver* graphique « nouveau » dans la liste noire du système. L’installation des outils de *build/compilation* assez standard.

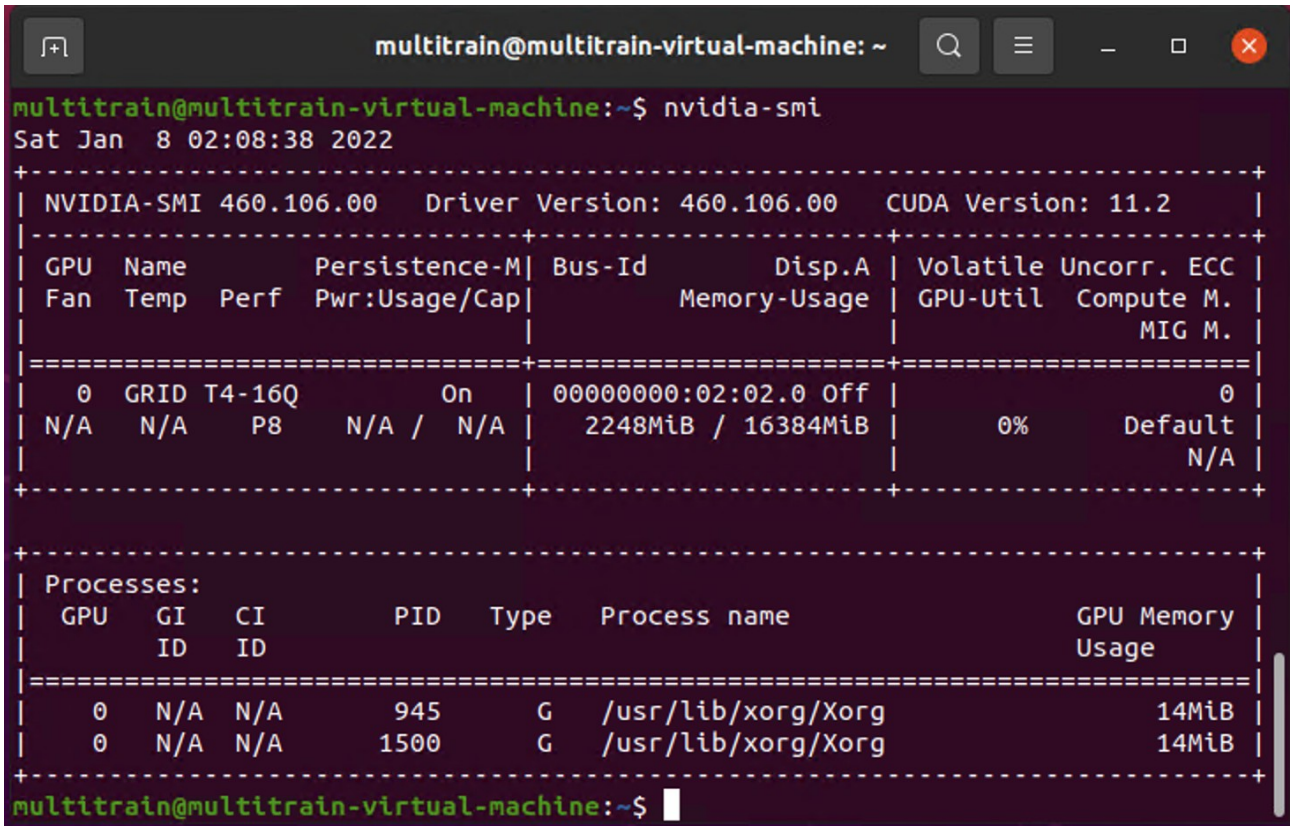

Figure 4: Commande nvidia-smi permettant de voir l'état du GPU

L’agent Horizon s’installe également sans encombre, ainsi que le composant PBISO qui permet la jonction à l’annuaire Active Directory de l’Université.

Cette installation a été grandement facilitée par le travail réalisé en amont par les équipes de la plateforme FALL qui avait déjà préparé et configuré des images VDI sous Ubuntu Linux.

4.2.2 API VMware

Les outils VMware proposent différentes API & SDK permettant d’interagir avec les plateformes vCenter, vSphere ou encore Horizon. L’API REST étant arrivée plus tardivement, nous avions déjà développé pour WINTER des scripts en PowerShell et des programmes utilisant le SDK pour Java afin d’automatiser un certain nombre de tâches.

Pour la plateforme FALL, nous avions déjà codé un certain nombre de scripts PowerShell utilisant les libraires PowerCLI de VMware et VimAutomation pour Horizon.

Ces bibliothèques sont relativement bien documentées et nous permettent de réaliser les scripts de cohabitation FALL / CIMENT.

Les contenus des codes sont documentés aux chapitres 4.3 - Pilotage des pools utilisateurs et 4.4 - Pilotage du pool de calcul.

4.2.3 Monitoring

Les premiers scripts réalisés en PowerShell avaient pour but de monitorer les indicateurs d'utilisation de la plateforme. Suivi des consommations CPU, RAM et IOPS sur le stockage, mais aussi suivi mémoire et processeur des cartes GPU. Côté Horizon, nous relevons les statistiques d'utilisation des différents *pools* de bureaux.

Figure 5: Exemple de tableau de bord de monitoring

Pour réaliser ces tableaux de bord, nous avons utilisé la pile logicielle suivante :

– PowerShell pour les scripts d'interrogation des API VMware ;

– Telegraf pour l'appel régulier des scripts Powershell et l'envoi au format InfluxDB ;

– InfluxDB pour stocker les données temporelles ;

– Grafana pour interroger la base InfluxDB et afficher des tableaux de bords.

4.3 Pilotage des pools utilisateurs

Comme nous l'avons expliqué précédemment, l'usage de la plateforme par des utilisateurs en dehors des heures de cours est rapidement devenu une habitude. C'est en effet, un des arguments en faveur des plateformes VDI (même si ce n'est pas la seule solution permettant de répondre à ce besoin).

Il était dès lors important de laisser une possibilité aux étudiants de travailler sur des logiciels, même ceux qui nécessitent des ressources GPU. L'utilisation des pools « classiques » (qui n'utilisent pas de ressources GPU) n'est pas affectée par la cohabitation FALL/CIMENT. On se concentre donc en particulier sur les pools GPU.

La première tentative consiste à désactiver les pools GPU utilisateurs entre 0h00 et 6h00. Et parmi ces pools GPU, nous gardons la possibilité d'exclure certains d'entre eux de ce processus.

Ci-après, un extrait du code exécuté à minuit.

```
#Cet extrait de code est volontairement simplifié pour servir le but didactique de cet article.

#On récupère d'abord l'ensemble des pools de la plateforme.
$pools = Get-HVPool

#On définit un tableau des pools que l'on souhaite exclure du processus d'extinction.
#On remarque que le pool de calcul 'fall-ciment-gpu' fait également partie de cette liste.
$pools_a_conserver = @('PoolGSv', 'fall-ciment-gpu')

#On parcourt ensuite les pools de manière itérative
foreach($pool in $pools)
{
 #Si le nom du pool ne fait pas partie des pools à conserver alors on continue
 if( $pools_a_conserver -inotcontains $pool.Base.Name )
 {
  #Si le pool porte la signature de l'utilisation de GPU alors on continue
  if($pool.DesktopSettings.DisplayProtocolSettings.PcoipDisplaySettings.EnableGRIDvGPUs)
  {
```

Une fois identifiés les pools que l'on souhaite cibler, nous allons effectuer 3 actions.

1. Désactiver le provisionnement automatisé des machines pour ce pool. C'est ce mécanisme qui génère en permanence des machines neuves disponibles pour les connexions des utilisateurs ;
2. Désactiver les accès au pool afin d'empêcher toute nouvelle connexion. Par contre, les connexions déjà en cours sont conservées et ne sont pas impactées ;
3. Supprimer les machines virtuelles du pool. En effet, les deux actions précédentes bloquent la création de machines et la connexion des utilisateurs, mais rien n'est fait pour les machines existantes. Or comme nous l'avons vu au chapitre 2.4 - Spécificités des GPU, les VMs GPU allumées bloquent des ressources système et vont empêcher le démarrage des VMs de calcul.

```
#On désactive le pool ainsi que son provisionnement automatique
$pool | Set-HVPool -Disable
$pool | Set-HVPool -Stop

#On récupère la liste des machines associées au pool
$machines = Get-HVMachine -PoolName $pool.Base.Name

#On parcourt ces machines
foreach($machine in $machines)
{
 #On s'assure de ne cibler que les machines qui ne sont PAS dans l'état CONNECTED
 #Les machines en état CONNECTED correspondent à un utilisateur final connecté.
 if ($machine.Base.BasicState -ne "CONNECTED")
 {
  #On supprime la machine sans autre forme de procès.
  Remove-HVMachine $machine.Base.Name -DeleteFromDisk -Confirm:$false
 }
}
```

Les VMs créées par Horizon et non associées à un utilisateur connecté peuvent être dans plusieurs états :

– disponible pour une connexion d'utilisateur ;

– en cours de provisionnement, si cette dernière vient d'être créée ;

– en cours de personnalisation, avant de passer à l'état disponible ;

– en maintenance, c'est-à-dire déjà utilisée une fois, et avant de passer à l'état Suppression ;

– en suppression, c'est-à-dire que vous avez compris je pense…

Dans tous les cas listés ci-dessus, la machine peut être supprimée sans précaution aucune, car :

– soit elle n'a encore jamais été utilisée ;

– soit elle l'a déjà été, et va être supprimée automatiquement par Horizon.

Elles ne contiennent donc aucune donnée critique.

Pour le script matinal, à 06h00 du matin, on réalise l'opération en chemin inverse.

```
$pools = Get-HVPool
$pools_a_conserver = @('PoolGSv', 'fall-ciment-gpu')

foreach($pool in $pools)
{
 if( $pools_a_conserver -inotcontains $pool.Base.Name )
 {
  if($pool.DesktopSettings.DisplayProtocolSettings.PcoipDisplaySettings.EnableGRIDvGPUs)
  {
   $pool | Set-HVPool -Enable
   $pool | Set-HVPool -Start
  }
 }
}
```

Une fois les accès au pool rétablis et le provisionnement automatique réactivé, Horizon se charge de recréer le nombre de VMs attendu sur la plateforme.

4.4 Pilotage du pool de calcul

Vous avez déjà plus ou moins compris que le pool de calcul est piloté en miroir des autres pools GPU. Allumage à 00h00, extinction à 06h00.

Toutefois, il convient d'ajouter 2 paramètres d'ajustement.

4.4.1 Temporisation de démarrage

Lorsque l'ordre d'extinction est donné aux pools GPU utilisateurs, suivi de l'ordre de suppression des machines virtuelles de ce pool, il s'écoule un certain délai avant que les machines soient effectivement supprimées et les ressources associées libérées.

Pour rappel, il est incontournable de libérer effectivement les ressources GPU, car le surbooking n'est pas supporté. Il convient donc de temporiser le démarrage du pool GPU et de s'assurer de la libération préalable de ces ressources.

Deux approches possibles, soit de manière empirique, attendre un certain délai de quelques minutes, en vérifiant que cela représente le délai réel constaté, soit patienter et vérifier que les machines sont effectivement supprimées. Nous avons opté pour ce deuxième choix, avec un point de vigilance : si d'aventure, certaines machines ne se suppriment pas, cela va interrompre la procédure. Jusqu'à ce jour, nous n'avons pas rencontré de problèmes de ce type.

```
# On parcourt à nouveau la liste des pools à éteindre
# On récupère, pour chaque pool, la liste des machines
$machines = Get-HVMachine -PoolName $pool.Base.Name

# Tant que ce nombre est supérieur à 0, on patiente...
while($machines.Length -gt 0)
{
 Write-Output "  Encore $($machines.Length) machines à éteindre"
 Start-Sleep -Seconds 5
 $machines = Get-HVMachine -PoolName $pool.Base.Name
}
```

Une fois les ressources effectivement libres, on peut passer à la suite.

4.4.2 Dimensionnement du pool de calcul

Une autre question dont la réponse n'est pas prévisible est de savoir combien d'étudiants seront connectés à 00h00 sur des logiciels utilisant des ressources GPU et que nous laisserons travailler ?

La réponse à cette question détermine la quantité de ressources GPU effectivement disponible pour le pool de calcul.

Un des paramètres que nous n'avons pas encore évoqué ici est le mécanisme de répartition des machines utilisant des ressources GPU sur les différentes cartes disponibles.

Nous avons opté pour un mode de concentration, quand 4 VMs associées à un profil mémoire GPU de 4 Go démarrent, le système va tenter de concentrer ces 4 VMs sur une seule carte physique, plutôt que de répartir 1 VM par carte physique (chaque serveur contient 4 cartes de 16 Go).

Les usagers ne sont pas impactés par cette mutualisation de carte, car on voit que les processeurs graphiques sont peu utilisés en termes de puissance de calcul. Ce constat étant posé, et sachant qu'une carte ne peut héberger que des profils mémoires identiques, ce mode nous semble être le plus adapté.

Dans le mode inverse, on peut imaginer un scénario pessimiste : 4 VMs associées à un profil mémoire GPU de 4 Go se répartissent sur les 4 cartes du serveur. Une VM associée à un profil mémoire GPU de 2 Go ne pourrait pas démarrer sur ce serveur, alors que les ressources mémoire GPU sont utilisées à 25 %

Les VMs du pool de calcul sont associées à un profil mémoire GPU de 16 Go. Elles ont donc besoin de cartes GPU complètement libres pour démarrer ! Cela renforce notre choix initial de concentrer plutôt que d'étaler les VMs GPU utilisateurs.

On arrive donc au dimensionnement du pool de calcul. On configure, dans Horizon, le nombre de machines attendu. Le système se charge ensuite de démarrer le nombre de VMs attendues. Nous avons fait le choix de viser le maximum, soit 40 VMs de calcul. La probabilité que les 40 cartes ne soient pas libres étant non nulle, nous avons dû tester le comportement du manque de ressources.

Le service Horizon va tenter de démarrer le nombre de machines attendu. En cas d'erreur répétée, il finit par abandonner et afficher une erreur au niveau du pool. Ce comportement ne gêne pas malgré tout le fonctionnement normal des calculs.

4.5 Perspectives d'évolution

Ce fonctionnement jour/nuit était une première itération. Nous avons envisagé plusieurs évolutions à apporter à ce modèle.

Première piste, on constate malgré tout que même dans la journée, une part des ressources GPU reste inutilisée. L'idée serait alors de pouvoir allumer et éteindre dynamiquement des VMs de calcul au cours de la journée. Cette piste a l'avantage de chercher à utiliser au maximum les temps morts.

Deuxième piste, autant des étudiants peuvent être encore connectés à minuit, autant à 3 h du matin, cela relève de l'exception ! Ne pourrait-on pas alors ajouter des VMs de calcul au cours de la nuit si des ressources deviennent disponibles ?

Enfin, du point de vue calcul, comment envisager que les ressources soient variables dans le temps, avec des VMs de calcul qui seraient disponibles uniquement pendant des créneaux courts de 15/30/45 min ? Est-ce vraiment utile ?

5 Bilan

Il est temps de dresser un premier bilan de cette collaboration entre les deux plateformes. Plusieurs aspects sont alors considérés.

5.1 Retours des utilisateurs VDI

Le contrat de départ avec les utilisateurs consistait à ne pas interrompre les besoins de VDI sans GPU et de ne pas interrompre les sessions VDI/GPU en cours. Avec le créneau 00h00-06h00, l'impact est nul sur les enseignements, et la possibilité de travailler en décalé reste possible.

Le tout reste donc assez transparent pour les utilisateurs finaux. De plus, cette collaboration entre FALL et CIMENT permet d'optimiser les répartitions des coûts de refacturation des cartes GPU et des licences associées. À la clé pour les utilisateurs VDI, c'est donc une diminution de la facture avec une qualité de service très peu voire non dégradée.

5.2 Retours des utilisateurs Calcul

5.3 Statistiques

La journée, on note que les VMs GPU se regroupent jusqu'à saturation des cartes. A minuit, on note un passage bref par le 0 % d'occupation qui correspond au moment où les pools GPU utilisateurs sont arrêtés. Dans la foulée, le pool de calcul prend le relais et occupe à son tour la mémoire des cartes.

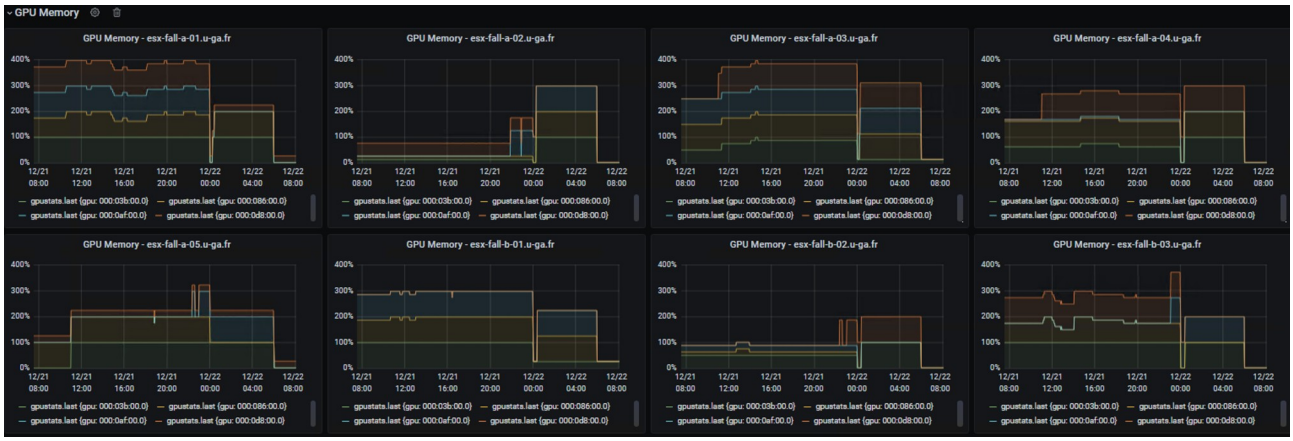

Figure 6: Profil d'utilisation mémoire GPU sur une journée et une nuit

Mais si l'on regarde l'utilisation en termes de processeur GPU, on note que les profils sont très différents. Les pools GPU utilisateurs occupent de la mémoire mais ne sollicitent que timidement les processeurs GPU. Cette utilisation minime s'explique par l'utilisation non intensive des applications exécutées.

Par contre, la nuit, on remarque une saturation franche des capacités de calcul GPU, qui diminue quand les jobs de calcul sont terminés.

Figure 7: Profil d'utilisation processeur GPU sur une journée et une nuit

5.4 Impact ROI plateforme VDI

De notre point de vue d'hébergeur, il est toujours intéressant d'augmenter le taux d'utilisation des ressources. Notre rôle est de trouver le bon modèle économique équilibrant au mieux les investissements réalisés et les refacturations du service aux entités qui l'utilisent.

5.5 Impact plateforme HPC

Du point de vue de CIMENT, la solution fonctionne comme attendu, ouvrant également quelques perspectives :

– Rendre dynamique la mise à jour de la date de fin de disponibilité ! Cela permettrait aux machines d'être rendues disponibles selon un calendrier variable. En effet, le gestionnaire de ressources a besoin de savoir à partir de quelle heure les ressources ne seront plus disponibles, afin de ne pas envoyer de jobs qui ne pourraient pas se terminer avant la disparition de celles-ci. Pour cela, il faudrait que l'on puisse récupérer dans l'OS des VM les informations du calendrier, afin que la ressource puisse indiquer au gestionnaire le timestamp correspondant à son heure d'extinction.

– Mettre en place un monitoring énergétique afin de pouvoir comparer la part de la consommation électrique d'une VM dédiée à un calcul sur GPU T4 avec celle d'un calcul sur GPU V100 ou A100 au sein d'un serveur physique. Il conviendra de ne pas uniquement se concentrer sur la consommation énergétique de la GPU, mais d'inclure aussi la part de consommation du serveur hôte (carte mère, mémoire centrale, cpu central, disques…). Le but étant de voir si la relative « lenteur » des T4 n'induit pas une surconsommation d'énergie du fait de la durée plus importante des jobs de calcul. Cette comparaison doit être faite sur plusieurs types d'applications afin de déterminer celles à éviter si elles engendrent un gaspillage énergétique sur ce type de plateforme.

6 Le mot de la fin

L'UGA expérimente régulièrement des collaborations entre les différents services de site. Le cluster VDI FALL étant le dernier arrivé dans la liste, c'était aussi la première collaboration pour l'équipe en charge de la plateforme. Grâce à l'expertise de chacun, l'architecture et la configuration de la solution se sont déroulés sans accroc. Les premiers retours sont positifs autant du côté des utilisateurs VDI que des utilisateurs calcul. Nous avons encore plusieurs pistes d'évolution pour affiner notre modèle, mais nous pouvons d'ores et déjà affirmer que cette collaboration est un réel succès !

6.1 Bibliographie

[1] Julien Tomassone - Rémi Cailletaud - Patrick Fulconis - Patrice Navarro - Christian Seguy
ACI, la solution SDN du datacentre de la Communauté Université Grenoble Alpes
JRES 2017 Nantes - https://2017.jres.org/fr/presentation?id=85

[2] Dimitri Rapacchi - Guenael Sanchez - Fabien Drago-Rajon - Manuel Estevez - Anthony Defize
SUMMER : 4 ans de stockage mutualisé et réparti
JRES 2017 Nantes - https://2017.jres.org/fr/presentation?id=72

[3] Guenael Sanchez - Gilles Bruno - Pascal Praly
WINTER : virtualisation hyperconvergée Full-Flash multi-site (retour d'expérience sur 1 an)
JRES 2017 Nantes - https://2017.jres.org/fr/presentation?id=70

[4] Comparaison des modalités d’accès aux GPU en environnement virtualisé
https://blogs.vmware.com/performance/2020/01/vmware-vsga-for-content-rich-vdi.html

[5] Bruno Bzeznik
OAR : Un gestionnaire de ressources pour grandes grappes de calcul
JRES 2009 Nantes - https://2009.jres.org/soumission/papers/render/pdf/39.pdf