Projet de Logiciel Cluster

Mise en place d'un supercalculateur dans un centre informatique.

Binôme : Matthias LAPU
Louis AUFFRET

Tables des matières

0 Introduction

Lors de ce projet, il nous était demandé d'analyser un appel d'offres émis par une université afin de déterminer si la société répondant à l'appel d'offres a réellement pris en compte tout ce que l'université a demandé. Nous avons remarqué que l'appel d'offre n'a pas fixé de budget, et dans la réponse d'offre, nous avons trouvé de nombreuses incohérences.

Par exemple, dans la page racking situé à la page 6 du sujet, il est question de 2 ports externe 1Gb dans les châssis AZ-B30, cependant, 1 seul port est évoqué dans le paragraphe sur les connexions externes, on a alors supposé la présence de 2 ports. Dans la synthèse de la réponse, située page 5 du sujet, il est noté pour la partition AX, "1 disque de 1 To", cependant, à la fin de la phrase, il est marqué "Aucun disque", on a supposé ces serveurs diskless. Ou encore, laissant place à nos suppositions, nous pouvons prendre l'exemple de la partie racking, ou il n'est pas précisé la taille physique des nœuds de login. On supposera ici qu'ils mesurent 2U. De même, il est question dans le sujet de relier les noeuds de login par 1 port 10 Gb au réseau backbone et 2 câbles 10 Gb au réseau d'administration, alors qu'ils n'ont que 2 ports 10 Gb.

Ces quelques incohérences, nous ont fait questionner si les erreurs que l'on trouvait dans le sujet étaient involontaires, ou si elles étaient là pour nous faire remarquer qu'AzkarHPC a fait quelques erreurs dans sa réponse.

1 Conformité de la réponse technique au cahier des charges

1.1 Partition AmdCpu (AC)

Dans le cahier des charges, il est demandé d'avoir environ 40 000 coeurs, la solution apportée par AzkarHPC comporte 38 656 coeurs, le nombre de coeurs est en dessous de la demande originelle mais est dans la marge des ±5%.

N'étant pas certains de l'avantage fourni par les processeurs hyperthreadés, nous nous sommes inspirés de clusters existants : en prenant un grand centre de calcul tel que le TGCC, nous pouvons voir que celui-ci utilise des processeurs hyperthreadés (ex : AMD Irene ROME qui possède des processeurs avec 64 coeurs, 128 threads et 256 MB de cache). C'est également le cas pour le choix effectué par le fournisseur. L'utilisateur n'est pas obligé d'utiliser l'hyperthreading si son code n'en profiterait pas, il semble donc cohérent d'en fournir le support.

Le fournisseur semble avoir fait un bon choix, en effet, dans la gamme de processeurs proposés, le seul ayant plus de cache L3 que les autres est le 9634 et celui-ci ne possède que 3/4 des coeurs du 9754 pour un prix proche. De plus, le 9754 est le processeur fournissant le plus grand nombre de coeurs, et il est hyperthreadé, ce qui permet d'améliorer les performances des codes conçus pour en profiter, il est donc un choix judicieux dans le cadre des codes CPU-bound, pour lesquels cette partition se doit d'être adaptée.

Pour ce qui est de la mémoire, AzkarHPC propose 256Go de mémoire par noeud, un noeud possède 128 coeurs, ce qui correspond donc a 2Go par coeur. Cela respecte la demande de l'université. Il n'y a pas de disques sur cette partition, car ils ne sont ni demandés ni nécessaires : AzkarHPC a fait le choix de boot les noeuds en PXE pour faire des économies.

L'un de leurs points forts réside dans leurs choix de processeurs, un processeur avec beaucoup de coeurs permet de réduire le nombre de noeuds nécessaire pour atteindre le nombre de coeurs demandé. En effet, s'ils avaient décidé d'opter pour le processeur 9634 qui possède le plus de cache L3, il aurait fallu plus de 475 noeuds pour avoir le même nombre de coeurs qu'avec leur proposition, et un code utilisant un plus grand nombre de noeuds est sujet à des effets NUMA inter-noeuds plus forts.

Cependant, avoir des coeurs trop nombreux peut ne pas être profitable, car il faudrait des utilisateurs capables d'utiliser 100% d'un processeur avec 128 coeurs et 256 threads, or d'après la loi d'Amdahl, pour pleinement utiliser cette parallélisation il faudrait avoir un code proche de 95% parallèle pour avoir un speedup théorique de 18, cependant en pratique cela est extrêmement dur à atteindre et les utilisateurs ne feront pas tourner de code utilisant autant de threads. De plus, augmenter le nombre de coeurs alloués peut impliquer d'augmenter le nombre de contrôleurs mémoire alloués, ce qui engendre des effets NUMA intra-noeud et peut dégrader les performances.

Les 302 lames peuvent être hébergées dans 11 châssis, un rack peut héberger 3 châssis, donc ces noeuds peuvent bien être hébergés dans 3 racks.

1.2 Partition AmdLarge (AL)

Le nombre de coeurs demandé dans l'appel d'offre est de 4000, AzkarHPC propose 4200 coeurs, ils sont au dessus de ce qui était demandé, mais dans la marge des ±5%.

AzkarHPC propose 50 noeuds, avec 512 Go de mémoire par noeud, pour 84 coeurs. Ce qui revient

à environ 6 Go par coeur de mémoire, ce qui est conforme à ce qui est demandé, et même au-delà. Notons que cette partition a pour but de faire tourner des codes memory-bound (à priori, vu la demande en mémoire), et le choix du processeur (9634) est particulièrement judicieux car c'est celui qui possède la plus grande quantité de cache L3, ce qui permet d'accélérer les accès mémoire plus souvent.

La demande de l'université n'est pas assez claire, elle ne précise pas les quantités de cache des CPUs, une réponse avec très peu de cache serait conforme à la demande, mais aurait des performances moindres sur des codes memory-bound. AzkarHPC n'a pas pris de risque et a proposé un cache L3 le plus important possible. Cependant, il aurait été possible de prendre un autre CPU, tel que le 9754 qui possède plus de coeurs, mais celui-ci possède moins de cache, donc le ratio cache/coeur est beaucoup plus faible, ce qui explique le choix du 9634.

En effectuant un changement de CPU, pour des 9754, avec 32 noeuds, il y aurait 4096 coeurs, et 134 400 $ économisés sur les CPUs uniquement.

Au total, avec moins de cache L3 (car nous ne savons pas si c'est que l'université désire réellement), nous pouvons faire une économie de plus de 134 000$ tout en ayant un nombre de coeurs élevé et 4 Go de mémoire par coeur, mais des performances mémoire moindres, en respectant malgré tout le cahier des charges. L'université n'ayant pas spécifié de budget, nous gardons ici la réponse proposée par AzkarHPC, mais en cas de réponse défavorable de l'université pour cause de budget, nous gardons en tête qu'il est ici possible de faire des économies.

Les 50 lames peuvent être hébergées dans 2 châssis, ce qui peut bien rentrer dans 1 rack. Cependant, nous avons fait le choix de mutualiser les emplacements (châssis et racks) pour les partitions AC et AL pour économiser de la place et des ports sur les switches évoqués dans les parties ci-dessus. Ainsi, avec 302 noeuds AC et 50 noeuds AL, on remplit :

- 10 châssis avec 30 noeuds AC chacun
- 1 châssis avec 30 noeuds AL
- 1 châssis avec 20 noeuds AL et 2 noeuds AC

Ainsi les partitions AC et AL peuvent rentrer dans 12 châssis au lieu de 13, donc dans 4 racks au lieu de 5, ce qui nous fait économiser de la place et permet de garder un rack libre si on voulait étendre les partitions.

1.3 Partition AmdGpu (AX)

AzkarHPC propose 50 noeuds avec des processeurs 9224, leur proposition offre au total 1200 coeurs, ce qui est loin d'être suffisant. De plus, avec une carte par noeud, il y aurait donc 50 cartes GPU Nvidia, ce qui ne correspond également pas à la demande émise par l'université. La quantité de mémoire par coeur excède 10 Go, ce qui est bien au-delà de la demande et n'est à priori pas nécessaire. De plus, il est marqué un disque de 1 To (incohérence car ils sont dits diskless juste après), ce qui n'est pas demandé. Remarquons également qu'il est demandé 60 cartes Nvidia A100, seulement 50 sont proposées, ce qui est en-dessous de la marge des $\pm 5\%$.

En supposant qu'un noeud AX ne peut accueillir qu'une seule Nvidia A100 (les spécifications du noeud ne sont pas précisées), on optera pour 60 noeuds comportant chacun une carte Nvidia A100, un processeur 9454 (48 coeurs, 96 threads), et 128 Go de RAM DDR5. On aurait alors 2880 coeurs,

2.67 Go de mémoire par coeur, et 60 cartes Nvidia : ce qui est conforme à la demande. Ces 60 noeuds peuvent remplir exactement 2.5 racks, on n'a pas besoin de plus que les 3 racks fournis.

Si un noeud pouvait accueillir 2 cartes, on opterait pour 30 noeuds ayant chacun 2 cartes Nvidia A100 (on remplirait à peine plus d'un rack), un processeur 9734 et 256 Go de DDR5 : on aurait alors 60 cartes Nvidia, 3360 coeurs et 2.29 Go de RAM par coeur, ce qui serait conforme à la demande (la demande spécifiant au moins 3000 coeurs, et non 3000 ±5%).

Si le budget proposé par l'université est dépassé, les modifications que nous avons proposé sur la partie AL seraient alors effectuées. En effet, Le budget originel proposé par AzkarHPC pour les CPU est de 91 250 € pour cette partie, si nous ajoutons la somme théorique dû au économies faites lors de la partie précédente, nous avons alors un nouveau budget de 230 450 $. Cependant nous ne connaissons pas les prix des cartes Nvidia ni des cartes mères, seulement ceux des processeurs.

1.4 Partition frontale (F)

La société propose 10 noeuds de login, ce qui est le nombre demandé par l'université. Chaque noeud possède un 9634, ce processeur possède 84 coeurs, il en était demandé 24, c'est donc bien au-delà de la demande. De plus, cela ne laisse pas assez de mémoire pour chaque coeur (1,14 Go).

Il faudrait par conséquent changer de processeur, ce qui ferait gagner en budget. Le coût pour 10 noeuds est de 103 040 $. Or, il est demandé 24 coeurs par noeud, nous pouvons donc utiliser le processeur 9224. Avec 10 noeuds 9224, cela reviendrait à 18 250 $, cela ferait économiser une somme importante (84 790 $), qui serait mieux utilisée ailleurs (partie AX). Notons par ailleurs que parmi les processeurs proposés, le 9224 a le meilleur rapport théorique GFLOPS/$ (en supposant que les FLOPS théoriques se calculent par $nb\_coeurs \times frequence$, sous les hypothèses qu'une opération sur des flottants dure un cycle, et qu'un code utilisant le processeur 100% du temps ne profite pas de l'hyperthreading, d'où le choix du calcul avec le nombre de coeurs physiques, bien que le 9224 supporte bel et bien l'hyperthreading).

Avec cette modification, la quantité mémoire originelle (96Go de mémoire DDR5) est trop élevée. En effet, cela reviendrait à avoir 4 Go par coeur, l'université n'en demande que 2. On optera ici pour 64 Go de RAM par noeud, ce qui fait 2.67 Go de RAM par coeur.

Le stockage requis ici est pour une partition /tmp et une partition bootable qui contient le système, les compilateurs, les bibliothèques de calcul, les debuggers, etc. La partition système peut être montée en read-only. La partition /tmp est montée en read-write. On a besoin de 3 To au total, on peut donc choisir 3 disques de 2 To et utiliser le contrôleur RAID pour avoir 2 disques de stockage (donc 4 To) et un de redondance (basé sur la parité, disque 3 = disque 1 XOR disque 2).

Le point fort de leur offre est le nombre important de coeurs, le point faible est le prix, qui est plus important que le nombre nécessaire pour des noeuds de login, et le controlleur RAID qui peut être mieux utilisé. De plus, la RAM disponible est également moins importante que la demande de l'université. Changer le processeur utilisé ainsi que la taille de la RAM permettrait de faire des économies et de respecter la marge de 5% de l'université.

1.5 Débit vers stockage Lustre

La société propose de mutualiser les 2 fonctions Lustre et Backbone sur 9 routeurs I/O. Pour la fonction lustre, ils seront connectés à l'aide de port IB HDR200 de part et d'autre. Il y a 9 routeurs

qui sont connectés.

Chaque routeur à un débit réel de 24 Go/s, en parallèle cela fait donc un débit théorique total de 216 Go/s. Cela dépasse de loin le débit demandé originellement, cependant, l'université indique "au moins", nous garderons donc la proposition d'AzkarHPC. L'un des points forts de cette offre repose dans le nombre de routeurs conséquent, permettant de proposer beaucoup plus de débit que demandé vers le stockage Lustre, ce qui peut s'avérer nécessaire en cas de panne ou de besoin d'extensibilité. De plus, la société propose les processeurs les moins chers, ce qui est un bon choix, car tous les processeurs possèdent le même débit mémoire, qui est amplement suffisant. Le fait d'avoir 2 processeurs par noeud permet également de répartir correctement la charge de travail, par exemple entre un processeur s'occupant uniquement du backbone et l'autre uniquement de Lustre.

On suppose que la raison pour laquelle il y a 2 processeurs par noeud est de fournir 1 par fonction (Lustre et Backbone) : cela pourrait permettre un meilleur usage du cache pour chaque processeur, car une interaction sur la mémoire par un processeur n'affectera pas le cache de l'autre tant que les zones mémoire utilisées sont distinctes.

1.6 Débit vers backbone

L'université demande 7.5 Go/s vers le backbone pour chacune des 3 partitions de calcul, AzkharHPC connectera 2 ports au backbone, si cela est bien configuré il est possible que cela double le débit. Au total 18 connexions 10 Gb seront établi, le débit sera donc entre 9 Go/s et 18 Go/s. Le point positif de leurs solutions repose dans la rapidité donnée par la double connexion des routeurs au backbone et le fait que le pire des cas apporte un meilleur résultat que ce que l'université demande.

1.7 Partition Admin/Service

La solution apportée par Azkhar nécessite 20 nœuds, cela est plus de nœuds que demander par l'université. Leurs solutions impliquent 2 nœuds maîtres permettant l'administration et le déploiement du cluster, ces 2 nœuds possède 2 disques de 3 To, nous supposons que ces disques contiendront les données nécessaires au déploiement. Ces 2 nœuds maîtres utiliseront également les données contenues dans les 2 disques de 3To des nœuds services pour stocker les données liées aux services internes. L'université demandait au total, au moins 80 To pour les données systèmes. AzkharHPC propose de séparer ces données dans différents nœuds, en ne suivant pas le format système/données systèmes demandés par l'université. Cependant, la partie maîtres et service seraient équivalents à la partie donnée systèmes, et la partie routeur, à la partie système. La solution proposée par Azkhar ne possède donc pas suffisamment d'espace par noeud pour correspondre à la demande de l'université. Cependant, leurs approches, utilisant des nœuds maîtres et des nœuds de services, permettraient, selon moi, d'avoir le résultat escompté avec moins d'espace, car un espace conséquent serait économisé avec une approche nœuds de services/neouds maîtres.

1.8 Offre logicielle & Outils de développement et de profiling

La société propose l'offre logicielle demandée par l'université, c'est-à-dire RHEL9. De plus, il y a également des rpm lustre, des compilateurs Intel et Amd. Uniquement, des processeurs Amd sont utilisés, nous supposons que le compilateur Intel sert uniquement afin d'avoir de la compatibilité si certains nécessitent des instructions assembleurs Intel. Sinon, ce compilateur est inutile. Il propose également des debugger ainsi que des logiciels Opensource. Ce qui est demandé par l'université.

2 Nommage du cluster et des composants matériels

2.1 Nom du cluster et Organisation logique des nœuds

Un nom doit être simple, et facile à retenir. L'université se nomme "Université Toulouse III - Paul Sabatier". **Nous appelerons donc le cluster "sabatier"**, c'est un nom simple et évocateur pour les personnes utilisant le supercalculateur de l'université.

Pour l'organisation logiques des noeuds, ceux-ci seront notés sabatierXXX. Dans la synthèse de la réponse, il y a déjà proche de 440 noeuds au total, dont 302, uniquement pour la partie AmdCPU. Afin de prendre en compte de futurs extension, nous utiliserons des plages assez grandes. Chaque groupe possédera son nombre afin de faciliter l'administration.

2.2 Définition des groupes clustershell

En s'inspirant du clustershell du cluster de l'ENSIIE. Nous appliquons la configuration suivante sur le fichier `/etc/clustershell/groups.d/cluster.yaml`.

```
login: sabatier[000-009]
master: sabatier[020-021]
services: sabatier[040-049]
io: sabatier[060-069]
nfs: sabatier080
ax: sabatier[100-149]
al: sabatier[200-249]
ac: sabatier[300-601]
all: sabatier[000-601]
```

Si l'on ajoutait en plus des serveurs NFS et des noeuds AX supplémentaires, on aurait par exemple sabatier[080-082] et sabatier[100-159] par exemple.

Pour la partie 4.3 et 4.4, nous avons compris qu'il était demandé de faire avec ce qu'AzkarHPC proposait, malgré le fait que lors de la 4.1 nous évoquions d'autres solutions, ou que des choses différentes étaient marquées dans l'énoncé (2 ports au lieu d'1 par exemple pour la partie AX 10 Gb etc.). Même si cela ne semblait pas toujours pertinent, nous avons préféré rester avec ce qu'il y avait dans l'énoncé afin de ne pas être pénalisés. Nous avons cependant pris le temps d'expliquer comment nos modifications auraient affecté ces questions.

3 Architecture du réseau de contrôle électrique 1Gb du cluster

Noeuds	Port Switch AZ-W40-1g	Commentaire
nœud master1 ports Gb [1-2]	ewc[1-2]p1	On élimine les SPOFs en le connectant à 2 switches.
nœud master2 ports Gb [1-2]	ewc1p2 ,ewc3p1	idem
1 serveur NFS admin, port BMC	ewc1p3	C'est un SPOF
9 nœuds de service, port BMC	ewc1p[4-6], ewc[2-3]p[2-4]	Réparti équitablement parmi les 3 switches
10 nœuds de login, port BMC	ewc1p[7-8], ewc[2-3]p[5-8],	2,4,4 . ewc1 possède déja les 2 masters, ont met plus de noeuds sur les 2 autres.
9 nœuds routeur, port BMC	ewc[1-3]	on réparti également équitablement
50 nœuds graphiques, port BMC	ewc1p[12-27], ewc[2-3]	on réparti à peu près équitablement (16,17,17)
Châssis AZ-B30, port BMC	ewc1p[28-35] ,ewc[2-3]p[29-36]	il y a 12 chassis, soit 24 port BMC, qu'on réparti équitablement
Les alimentations électriques (PDU)	ewc[2-3]p37	les 2 masters sont déja sur ewc1, on répartit les PDU sur les 2 autres

Utilisation des ports :

- Switch 1 : 35 ports utilisés, 5 ports restants
- Switch 2 : 37 ports utilisés, 3 ports restants
- Switch 3 : 37 ports utilisés, 3 ports restants

Finalement, lorsque tout est connecté, 109 ports sont utilisés et il reste 11 ports.

Ce nouveau branchement, rend le réseau de contrôle électrique 1Gb plus résilient, car tout est réparti de manière à peu près uniforme. S'il y a une panne de port, il reste suffisamment de port pour le déplacer et le rendre de nouveau fonctionnel en attendant de réparer le problème de port. S'il y a une panne de port et que celui-ci n'est pas déplacer, les fonctionnalités permettant d'allumer et d'éteindre la dite machine est donc perdu (nœuds de calcul, nœuds de service, etc.) . Sur le réseau de contrôle, les flux utilisés sont en lien avec la gestion de nœuds.

Il n'est pas précisé comment les switchs sont interconnectés entre eux pour former un switch virtuel. S'il est nécessaire de les relier entre eux en utilisant l'un des ports. 2 ports en plus, par switch seront utilisés pour faire un graphe complet. Il resterait donc : 3 ports pour le switch 1, et 1 port pour le switch 2 et 3.

Notons que la partition AX utilise à elle seule presque la moitié des ports, et y ajouter 10 noeuds pour se conformer à la demande de l'université implique qu'il n'y a plus assez de ports et qu'il faudrait rajouter un switch et revoir les attributions des ports pour mieux distribuer la charge.

Les 2 noeuds maîtres possèdent chacun 1 port 1 Gb relié au port BMC de l'autre, ainsi que 3 autres ports 1 Gb. Ici, en utiliser seulement 2 est suffisant pour éviter que la panne d'un switch empêche complètement l'accès au réseau électrique par l'un des noeuds maîtres, c'est donc la solution retenue.

Ceci dit, nous pensons qu'1 seul port BMC pour le serveur NFS Admin représente un SPOF conséquent, en effet, si le switch tombe en panne, les administrateurs perdent l'accès BMC au NFS

permettant de stocker les données du système nécessaires à la gestion du cluster. Celui-ci est censé rester allumé, mais par précaution nous avons pensé à une alternative. S'il était possible d'effectuer des modifications à la proposition d'AzkarHPC, nous aurions rendu la partie NFS résiliente ainsi :

Noeuds	**Port Switch AZ-W40-1g**	**Commentaire**
nœud master1 ports Gb [1-2]	ewc[1-2]p1	On élimine les SPOFs en le connectant à 2 switchs.
nœud master2 ports Gb [1-2]	ewc1p2 ,ewc3p1	idem
1 serveur NFS maitre proposé par Azkhar, port BMC	ewc1p3	On ajoute de la résilience avec un NFS par switch
1 serveur NFS maître supplémentaire , port BMC	ewc2p2	
1 serveur NFS esclave , port BMC	ewc3p2	
9 nœuds de service, port BMC	ewc1p[4-6], ewc[2-3]p[3-5]	Réparti équitablement parmi les 3 switchs
10 nœuds de login, port BMC	ewc1p[7-8], ewc[2-3]p[6-9]	2,4,4 . ewc1 possède déja les 2 masters, ont met plus de noeuds sur les 2 autres.
9 nœuds routeur, port BMC	ewc1p[9-11] , ewc[2-3]p[10-12]	on réparti également équitablement
50 nœuds graphiques, port BMC	ewc1p[12-27], ewc[2-3]p[13-29]	on réparti à peu près équitablement (16,17,17)
Châssis AZ-B30, port BMC	ewc1p[28-35] ,ewc[2-3]p[30-37]	il y a 12 chassis, soit 24 port BMC
Les alimentations électriques (PDU)	ewc[2-3]p38	les 2 masters sont déja sur ewc1, on réparti les PDU sur les 2 autres

4 Architecture du réseau d'administration 10 Gb du cluster

Nous rappelons qu'il y a 12 châssis.

Noeuds	Port Switch AZ-W40-10g	Commentaire
nœud master1 ports 10Gb [1-2]	ewa1p1, ewa4p1	On répartit équitablement
nœud master2 ports 10Gb [1-2]	ewa2p1, ewa5p1	idem
1 serveur NFS adm, 4 port 10Gb	ewa1p2, ewa2p2, ewa3p1, ewa4p2	idem
9 nœuds de service, 1 port 10Gb	ewa1p3, ewa2p3, ewa3p2, ewa4p[3-4], ewa5p[2-3] , ewa6p[1-2]	idem
10 nœuds de login , 1 port 10Gb	ewa1p4, ewa2p[4-5], ewa3p[3-4],ewa4p5, ewa5p[4-5], ewa6p[3-4]	idem
9 nœuds routeur, 1 port 10Gb	ewa1p5, ewa2p6, ewa3p5, ewa4p[6-7], ewa5p[5-6], ewa6p[5-6]	idem
50 nœuds graphiques, 1 port 10Gb	ewa1p[6-13], ewa2p[7-14], ewa3p[6-14], ewa4p[8-15], ewa5p[7-14], ewa6p[7-15]	idem
12 châssis AZ-B30, 4 ports 10Gb	ewa1p[14-21], ewa2p[15-22], ewa3p[15-22], ewa4p[16-23], ewa5p[15-23], ewa6p[16-23]	idem

Nous avons réparti tous les noeuds entre les différents switches de manière uniforme en utilisant du Round-robin. Cela nous permet d'éliminer les SPOFs en évitant que des ports équivalents (venant du même noeud ou de noeuds redondants entre eux) se retrouvent attribués aux ports d'un même switch, ce qui ferait du switch en question un SPOF

Utilisation des ports :

- Switch 1 : 21 ports utilisés, 19 ports restants
- Switch 2, 3 : 22 ports utilisés, 18 ports restants
- Switch 4, 5, 6 : 23 ports utilisés, 17 ports restants

Finalement, lorsque tous les noeuds sont connectés, il reste 106 ports libres.

Il faut également établir la topologie nécessaire à la création de ce switch virtuel. En effectuant une topologie cyclique, c'est-à-dire que le switch est relié au switch n+1 et n-1 modulo le nombre de switchs disponible. Avec cette topologie, une panne d'un des switches n'empêche pas les autres de communiquer entre eux, 146 ports seraient utilisés et il resterait 94 ports libres.

Si nous effectuons une topologie full mesh (graphe complet), 164 ports serait utilisés et il resterait 76 ports libres. Étant donné le nombre de ports restants qui est conséquent, nous pouvons partir sur une topologie full mesh, permettant donc un accès à chaque switch depuis 1 machine externe en au plus 2 switches.

Si un port tombe en panne, il restera assez de place pour le déplacer le temps de résoudre la panne. La perte d'un port entraîne la perte de l'accès au service lié à celui-ci en passant par ce port. Par conséquent, les noeuds n'ayant qu'un seul port ne seront plus administrable. Les types de flux/protocoles utilisés sont en lien avec les services d'administration, il y a également les nœuds qui auront besoin de faire des requêtes (mises à jour, puppet, etc.).

Si un switch tombe en panne, la résilience de notre solution permet aux autres de toujours fonctionner, cependant, cela réduit également l'administration des services du cluster d'1/6.

Remarquons que dans l'énoncé, il est marqué que les nœuds de services, de login, les routeurs et les noeuds AX sont connectés au réseau 10Gb par 2 câbles et non 1 comme indiqué dans le tableau. Il faudrait alors rajouter plus de câbles qu'indiqué dans le tableau en augmentant les tailles des plages allouées, il y a la place pour cela, et cela remplirait presque les switches. Cela permettrait d'ajouter plus de résilience et de permettre l'administration de chaque machine même si un port tombe en panne. Sauf pour les noeuds de login, qui n'ont que 2 ports 10 Gb, l'un d'eux étant réservé au réseau backbone. De plus, si nous prenons en compte la solution énoncée lors de la partie précédente il faudrait également rajouter des ports pour le serveur NFS admin rajouté, ainsi que le serveur NFS

esclave. C'est une solution plus viable que le tableau donnée dans l'énoncé. Et la solution que nous aurions utilisé s'il était possible d'effectuer l'architecture à partir de nos modifications fait à la partie 1.

5 Connectivité externe au Backbone de l'université

Le réseau backbone fournit des services LDAP, DNS, NTP, NFS et PXE, ainsi qu'une communication directe avec les stations de travail de l'université. Le LDAP sert notamment à l'identification et l'authentification. Le DNS sert surtout pour l'accès à Internet, car s'il y a besoin de noms de domaines pour certaines machines au sein du réseau, un simple fichier /etc/hosts suffit, donc le serveur DNS n'est accessible qu'aux administrateurs sur les noeuds maîtres. Pour choisir sur quel VLANs placer les noeuds, il est nécessaire de prévoir les services dont ils auront besoin :

- les noeuds de service auront accès au service NTP (strate n) et serviront eux-mêmes de service NTP (strate n+1) pour les noeuds au sein du réseau IB, et ils auront accès au service PXE : les noeuds diskless du réseau IB vont boot en PXE sur les noeuds de service, les noeuds de service vont soit aller chercher les images de boot sur le serveur PXE de l'université, soit les envoyer directement s'ils les ont en cache (le cache permet d'éviter de surcharger le PXE de l'université)
- les noeuds maîtres sont réservés aux administrateurs, ils ont accès à tous les services, et les stations de travail peuvent y avoir accès en SSH
- les noeuds de login ont besoin de LDAP pour l'identification et l'authentification, et de NFS pour monter les répertoires home des utilisateurs, un serveur SSH devra écouter sur le réseau des stations de travail
- les routeurs n'ont à priori besoin de rien sur le réseau backbone, car récupérer des contenus provenant du cluster Lustre peut se faire en passant par les noeuds de login, mais il serait intéressant d'héberger un serveur NFS sur ces routeurs pour avoir un accès direct aux contenus du cluster Lustre depuis les stations de travail.

Nodeset	**Serveurs université**	**Flux/protocoles**
@login	montcalm[1-2], maladeta, stations	LDAP, NFS, SSH
@master	montcalm[1-2], maladeta, ossau[1-2], marbore[1-2], perdu[1-2], stations	LDAP, NFS, NTP, DNS, PXE, SSH
@service	ossau[1-2], perdu[1-2]	NTP , PXE
@io	stations	NFS

Pour ce qui est du nombre de ports :

- les 9 noeuds de service (4 ports 10 Gb) ont déjà 1 port 10 Gb sur le réseau d'administration, il est envisageable d'en ajouter un 2e sur ce même réseau, donc on utilise les 2 restants pour le réseau backbone (18 ports à demander)
- les 2 noeuds maîtres (4 ports 10 Gb) ont déjà 2 ports 10 Gb sur le réseau d'administration, on utilise les 2 restants pour le réseau backbone (4 ports à demander)
- les 10 noeuds de login (2 ports 10 Gb) ont déjà 1 port 10 Gb sur le réseau d'administration, on utilise celui qui reste pour le réseau backbone (10 ports à demander)
- les 9 noeuds routeurs (6 ports 10 Gb) ont déjà 1 port 10 Gb sur le réseau d'administration, il est envisageable d'en ajouter un 2e sur ce même réseau, donc on utilise les 4 restants pour le réseau backbone (36 ports à demander)

Il faudra demander au total 68 ports sur le réseau backbone à l'université.

6 Architecture du réseau d'interconnexion Infiniband

Tous les équipements du cluster (sauf le NFS, qui n'est que sur le réseau d'administration) possèdent au moins un port InfiniBand. Les routeurs en ont deux, mais l'un d'eux est réservé au cluster Lustre. Le seul port InfiniBand disponible sur chacun des noeuds sera donc relié au réseau InfiniBand du cluster. La topologie proposée par AzkarHPC est en étoile, avec 6 switches L2 redondants au centre de l'étoile et 17 swicthes L1 comme branches, chacun connecté aux 6 switches L2. On a alors 17 ports de chaque switch L2 utilisés. Pour les noeuds des partitions AC et AL, les switches InfiniBand sont ceux des châssis (30 noeuds/L1), or on a 12 châssis AC/AL donc il faudrait répartir le reste sur 5 switches L1. AzkarHPC propose d'avoir des switches L1 dédiés pour la partition AX (il en faudrait alors 2 au vu du nombre de noeuds, avec 25-30 noeuds/L1), un switch L1 dédié aux noeuds de service (9 noeuds/L1) et un dédié aux noeuds de login (10 noeuds/L1). D'après la solution proposée, il ne reste qu'un seul switch L1 pour les routeurs et les noeuds maîtres (soit 11 noeuds/L1).

Cependant, même si cette solution est conforme en termes de nombre de ports utilisés, elle manque de redondance et de résilience : les 9 routeurs sont sur le même switch, or les routeurs font les I/O les plus intensives. De plus, si le switch en question ne fonctionnait plus, les noeuds maîtres n'ont plus accès au réseau IB, et le réseau IB n'a plus accès au cluster Lustre. Si le switch qui héberge les noeuds de login ne fonctionnait plus, aucun utilisateur n'aurait accès au réseau IB. Si le switch qui héberge les noeuds de service ne fonctionnait plus, le cluster entier se met à dysfonctionner.

Étant donné que chaque équipement ne possède qu'un seul port InfiniBand pour ce réseau, on ne peut pas introduire de la redondance en connectant un équipement à plusieurs switches L1 : les L1 seront des SPOFs quoi qu'il arrive. Cependant, on peut répartir sur plusieurs switches L1 les noeuds dont les I/O sont les plus intensives (les routeurs) et les noeuds redondants (noeuds de service, noeuds maîtres, noeuds de login utilisés par les mêmes groupes).

Nous proposons donc de garder les 12 switches dédiés à AC/AL, et les 2 switches dédiés à AX, mais de répartir le reste des noeuds sur les 3 switches restants :

Noeuds	Ports IB L1	Commentaire
Switch L2 x6	iwh1n[1-17]p[1-6]	chaque L1 relié à tous les L2
Routeur x9	iwh1n[1-3]p[10-13]	répartition de charge, redondance
Service x9	iwh1n[1-3]p[15-17]	redondance
Maître x2	iwh1n[1-2]p20	redondance
Login x2 (groupes ugp[0-2])	iwh1n[2-3]p21	redondance
Login x3 (groupes ugp[3-4])	iwh1n[1-3]p22	redondance
Login x2 (groupes ugp[5-7])	iwh1n{1,3}p23	redondance
Login x3 (groupes ugp[8-9])	iwh1n[1-3]p24	redondance

ici iwh1[4-17] sont dédiés aux partitions AC, AL et AX

Ainsi, si l'un des 3 switches ne fonctionnait plus, seul l'un des deux noeuds maîtres n'aura plus accès au réseau IB, chaque groupe d'utilisateurs aurait accès à au moins un noeud de login ayant accès au réseau IB, et le débit vers Lustre est réduit. Si 2 des 3 switches ne fonctionnait plus, soit un groupe d'utilisateurs n'a plus aucun accès au réseau IB, soit les 2 noeuds maîtres n'y ont plus accès, et le débit vers Lustre est très réduit. De plus, la charge des routeurs est répartie équitablement sur les 3 switches L1.

7 Extensibilité de la solution

Le schéma du réseau IB a déjà été revu lors de la question précédente.

7.1 Sans rajouter de châssis

Dans la situation actuelle, il y a 12 châssis, chaque châssis contient 30 lames. D'après l'appel d'offres, il y a actuellement 352 nœuds, 300 de la partition AC et 50 de la partie AL. Or, chaque lame peut contenir 30 nœuds, nous pouvons donc rajouter 8 nœuds sans rajouter de châssis.

7.2 Sans rajouter de racks

1 châssis fait 16U, 1 racks en fait 48U. Avec les 12 châssis de la partie AC+AL, nous remplissons 4 racks. D'après le nombre de racks indiqué dans l'appel d'offres, il reste donc 1 rack vide pour la partition AC+AL. En rajoutant des châssis, nous pouvons alors le remplir. Cela rajouterait donc 3 châssis, donc 90 nœuds supplémentaires. Par conséquent, sans rajouter de racks, mais uniquement en rajoutant des nœuds et des châssis, ils seraient possibles de rajouter 98 nœuds (90 nœuds de cette partie + les 8 nœuds de la partie précédente).

7.3 Facteurs limitant et extension de la partie AX

Le problème de la partie AX repose dans le fait qu'il n'y a qu'un GPU par nœud. Cela fait donc beaucoup de connexions aux différents switches. En effet, en comparant le nombre de nœuds AX et le nombre de nœuds de la partition AC+AL, nous pouvons voir que 50 nœuds AX occupent 50 ports sur chacun des réseaux du cluster (IB, administration, contrôle électrique), alors que 12 châssis avec 352 nœuds et la topologie proposée occupent 72 ports sur les switches (externes aux châssis) du réseau IB, 48 ports sur le réseau d'administration, et 24 ports sur le réseau de contrôle électrique. Il devient envisageable d'introduire des switches supplémentaires dans les racks de la partition AX, mais il y a une meilleure solution.

En optant pour des serveurs permettant de mettre plusieurs GPU dans un nœud tel que le "Serveur De Calcul APY SCG GPGPU 2U 4 GPU AMD EPYC Serie 7003", nous pourrions diminuer de manière conséquente le nombre de nœuds AX. Cela permettrait de mettre 4 cartes Nvidia A100 par nœud, augmentant grandement l'extensibilité sans rajouter de rack et sans augmenter de manière absurde le nombre de ports nécessaires.

Cependant, cela est coûteux et nécessiterait de revoir tous les nœuds AX et de refaire les branchements pour les switchs.

7.4 Limitation de l'extensibilité

Cependant, même en appliquant les modifications précédentes, si le budget n'est pas une limite pour l'université, le débit et la latence le seront. En effet, il faut des connexions à chaque nœud lié, notamment, à la topologie, car un full mesh n'est pas faisable si le cluster continue de s'agrandir. Et même des topologies tel que dragonfly poseraient problème passé un certain nombre de nœuds.

De plus, un problème de refroidissement serait à prévoir si nous utilisons les noeuds avec 4 GPU proposés précédemment et que nous remplissons des racks avec ceux-ci. Une autre limitation pourrait être un problème d'espace lié au bâtiment et au poids de chaque rack. Ce qui forcerait une modification de l'infrastructure du bâtiment contenant le calculateur, ou une limitation du nombre de noeuds par rack, amenant à acheter plus de racks. Ce qui représente un budget non négligeable, que ce soit pour une université ou non.

8 Récupération des logs et des consoles

8.1 Choix des nœuds et de l'architecture

Afin de pouvoir récupérer les logs ainsi que les consoles pour administrer le cluster, nous mettrons en place un serveur de logs centralisé. Les fichiers de journaux sont organisés en utilisant une arborescence permettant de distinguer facilement les machines qui ont généré les logs. C'est-à-dire /var/cluster/logs/ ⟨ Hostname ⟩ (comme demandé).

Pour des raisons de scalabilité, le serveur NFS est choisi pour stocker les logs en raison de sa capacité de stockage. D'après la configuration proposée par AzkarHPC, il y a un seul serveur NFS, ce qui en fait un SPOF, il est donc vivement encouragé de faire un serveur NFS redondant comme évoqué précédemment.

Le serveur NFS a pour rôle de stocker les logs. Il est tout à fait possible pour les nœuds de calcul d'envoyer directement les logs au NFS, mais il faudrait pour cela un démon rsyslog sur le serveur NFS qui gérerait la politique de décision (savoir dans quel fichier va quel log). Afin d'éviter de faire des capacités de calcul (inconnues) du serveur NFS un SPOF, la politique de décision est confiée aux nœuds de service. Aucun démon rsyslog ne tourne sur le NFS.

Les nœuds de service sont non seulement en charge de la politique de décision, mais aussi du logrotate (car la compression demande aussi des ressources CPU). Ainsi, les logs générés par tous les nœuds du réseau IB seront envoyés aux nœuds de service (via le réseau IB), qui se chargeront de décider où les stocker. Sur ces nœuds de service, plusieurs répertoires de logs sont montés en NFS, le nœud de service décide simplement où placer les logs, ils sont alors envoyés des nœuds de service au NFS via le réseau d'administration (d'où le choix d'envoyer les logs aux nœuds de service via le réseau IB, pour éviter de doubler le trafic sur le réseau d'administration).

Étant donné qu'il y a 4 switches d'administration reliés directement au NFS, nous décidons, pour des raisons de débit sur le réseau d'administration, d'attribuer la gestion des logs à 4 nœuds de service, un sur chacun de ces switches : sabatier[40-43].

Dans cette configuration, on peut cependant relever un problème : comme le logrotate compresse les logs à heure fixe, une grande quantité de logs est compressée à chacune de ces heures. Ce qui signifie qu'une grande quantité de logs est envoyée par le NFS aux nœuds de service, compressée sur ces nœuds de service puis renvoyée au NFS, ce qui pourrait saturer régulièrement le réseau d'administration.

Si les capacités de calcul du serveur NFS étaient connues et qu'elles s'avéraient suffisantes, il serait envisageable de lui attribuer la politique de décision et le logrotate, ainsi les nœuds de service n'auraient même plus besoin de servir d'intermédiaire, et les nœuds du réseau IB pourraient envoyer leurs logs directement au NFS via le réseau d'administration. Dans ce cas, les nœuds de service ne fourniraient qu'un accès en lecture au NFS. Ne connaissant pas la capacité de calcul du serveur NFS, cette solution n'est pas retenue. La solution retenue est celle évoquée dans les paragraphes ci-dessus, malgré le problème de réseau.

Les fichiers de configuration différeront sur les nœuds de calcul et les autres nœuds : tous les logs des autres nœuds seront enregistrés, seuls les logs évoquant un problème seront envoyés par les nœuds de calcul.

Les nœuds de services sont ceux qui répartissent les logs. Leurs configurations se situent dans le fichier rsyslog_service.conf. Rappelons qu'il y a différents niveaux d'urgence pour les logs, allant de 0 à 7 (0 étant le plus urgent et 7 le moins urgent). Notre politique de gestion des journaux du cluster

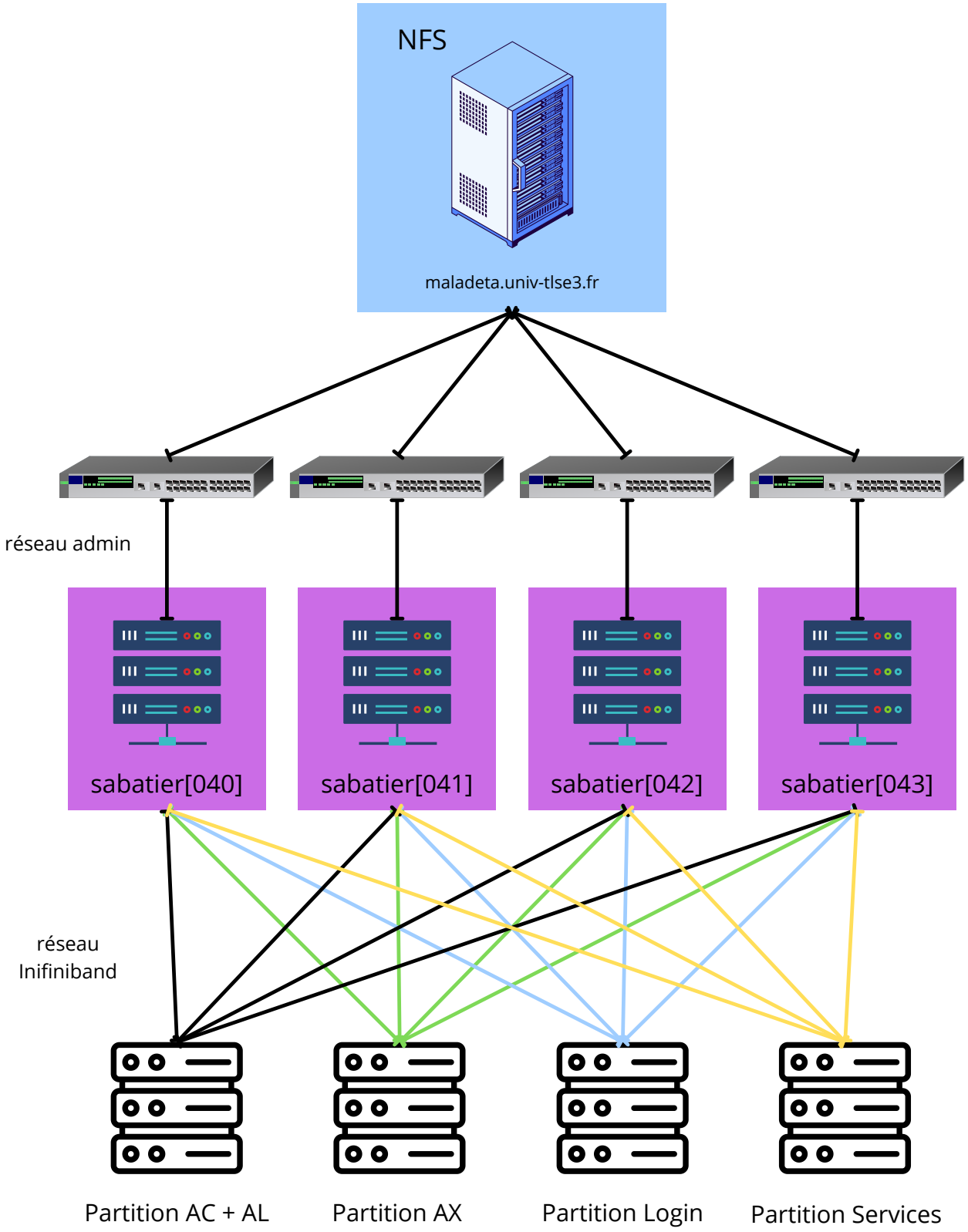
est donc la suivante (cf. logrotate.conf) :

- Les codes d'erreurs allant de 0 à 2 sont des erreurs critiques. Les problèmes critiques doivent être résolus rapidement. Cependant, nous pensons qu'il est important d'avoir accès à ceux-ci rapidement en cas de problème similaire/redondant sur une période de temps relativement longue. C'est pour cette raison que nous les stockerons 1 an. Ceux-ci seront "rotate" et compressé tous les mois. Ces logs sont censés être les messages les moins nombreux mais les plus importants.
- Les codes d'erreur de niveau 3 ne sont pas urgents, mais doivent être traités rapidement. Ils seront "rotate" et compressés tous les mois, et supprimés au bout de 6 mois. Nous mettrons également les logs en lien avec ldap ainsi que le boot dans cette catégorie. Ces messages peuvent également être utiles et ne devraient pas être trop nombreux.
- Les autres logs seront conservés 3 jours puis supprimés. Ce sont de loin les plus lourds, ils ne seront donc pas stockés longtemps.

Un template est également utilisé sur les logs afin que ceux-ci soient analysés par un logiciel de visualisation de logs tel qu'Opensearch, ou la version payante, Elasticsearch. Ce template a été écrit par le développeur de Rsyslog et est situé au début du fichier de configuration des nœuds de services.

Il y a également la configuration des nœuds de calcul et autres clients Rsyslog, celle-ci est située dans le fichier rsyslog_node.conf, un failover sera implémenté, le nœud client vérifiera donc si d'autres nœuds de service sont disponibles pour prendre le relais. De plus, nous mettons également en plus du load-balancing afin de ne pas surcharger une seule machine.

Afin de centraliser les consoles des nœuds afin d'avoir accès à l'ensemble des logs du cluster, nous pouvons utiliser le logiciel interactif "conman". Il permet de visualiser les logs de plusieurs machines en même temps.


NFS
maladeta.univ-tlse3.fr
réseau admin
sabatier[040]
sabatier[041]
sabatier[042]
sabatier[043]
réseau
Inifiniband
Partition AC + AL
Partition AX
Partition Login
Partition Services

8.2 Fichiers de configurations

Le fichier de configuration logrotate:

Listing 1: Configuration logrotate.conf

```
# ils seront compresses lors de leurs rotation (parametre par defaut)
compress

/var/cluster/logs/*/panic.log
/var/cluster/logs/*/alert.log
/var/cluster/logs/*/critical.log {
    rotate 12
    monthly
    missingok
    notifempty
    create 0644 root root
    endscript
}

/var/cluster/logs/*/error.log {
    rotate 6
    monthly
    missingok
    notifempty
    create 0644 root root
    endscript
}

/var/cluster/logs/*/ldap.log
/var/cluster/logs/*/boot.log
/var/cluster/logs/*/warning.log {
    rotate 1
    weekly
    missingok
    notifempty
    create 0644 root root
    endscript
}

/var/cluster/logs/*/messages{
    rotate 3
    daily
    missingok
    notifempty
    create 0644 root root
    endscript
}s
```

Le fichier de configuration pour les nœuds de service:

Listing 2: Configuration des nœuds de services rsyslog_service.conf

```
# this template is used to generate JSON output for elasticsearch (just in case)
# from a lead dev of rsyslog
# (https://rainer.gerhards.net/2018/02/simplifying-rsyslog-json-generation.html)
```

```
template(name="outfmt" type="list" option.jsonf="on") {
        property(outname="@timestamp"
                name="timereported"
                dateFormat="rfc3339" format="jsonf")
        property(outname="host"
                name="hostname" format="jsonf")
        property(outname="severity"
                name="syslogseverity-text" caseConversion="upper" format="jsonf")
        property(outname="facility"
                name="syslogfacility-text" format="jsonf")
        property(outname="syslog-tag"
                name="syslogtag" format="jsonf")
        property(outname="source"
                name="app-name" format="jsonf")
        property(outname="message"
                name="msg" format="jsonf")
    }
#### MODULES ####
# The imjournal module below is now used as a message source instead of imuxsock.
# provides support for local system logging (e.g. via logger command)
$ModLoad imuxsock
# provides access to the systemd journal
$ModLoad imjournal
#$ModLoad imklog # reads kernel messages (the same are read from journald)
#$ModLoad immark # provides --MARK-- message capability
# Provides UDP syslog reception
#$ModLoad imudp
#$UDPServerRun 514
# Provides TCP syslog reception
$ModLoad imtcp
$InputTCPServerRun 514
#### GLOBAL DIRECTIVES ####
# Where to place auxiliary files
$WorkDirectory /var/lib/rsyslog
# Use default timestamp format
$ActionFileDefaultTemplate RSYSLOG_TraditionalFileFormat
# Include all config files in /etc/rsyslog.d/
$IncludeConfig /etc/rsyslog.d/*.conf
# Turn off message reception via local log socket;
# local messages are retrieved through imjournal now.
$OmitLocalLogging on
# File to store the position in the journal
$IMJournalStateFile imjournal.state
#### RULES ####
# Log anything that is an avenger level threat.
*.panic /var/cluster/logs/$HOSTNAME%/panic.log
# Log alert
*.alert /var/cluster/logs/$HOSTNAME%/alert.log
# Log critical
*.crit /var/cluster/logs/$HOSTNAME%/critical.log
# Log anything else
*.debug /var/cluster/logs/$HOSTNAME%/messages.log
### DEFAULT (just changed log to cluster) ###
# The authpriv file has restricted access.
```

```
authpriv.* /var/cluster/logs/$HOSTNAME%/secure
# Log all the mail messages in one place.
mail.* -/var/cluster/logs/$HOSTNAME%/maillog
# Log cron stuff
cron.* /var/cluster/logs/$HOSTNAME%/cron
# Save news errors of level crit and higher in a special file.
uucp,news.crit /var/cluster/logs/$HOSTNAME%/spooler
# Save boot messages also to boot.log
local7.* /var/cluster/logs/$HOSTNAME%/boot.log
# LDAP
local4.* /var/cluster/logs/$HOSTNAME%/ldap.log
#### JOURNAL ####
*.* /var/cluster/logs/%HOSTNAME%/messages
#### CONMAN ####
if $programname == 'conmand' then {
    action(type="omfile" file="/var/cluster/logs/%HOSTNAME%/conman.log")
}
```

Notons qu'en fonction de la politique du centre ainsi que de la bande passante de l'UDP pourrait être préférable à du TCP pour ne pas surcharger le réseau si ce sont des logs qui peuvent être perdues sans que cela soit grave.

Le fichier de configuration pour les clients rsyslog :

Listing 3: Configuration des nœuds rsyslog_node.conf

```
# enable load balancing (https://www.rsyslog.com/doc/configuration/modules/omfwd.html)
# every 300 seconds, the next target is selected
$ActionSendTCPRebindInterval 300

*.* @@sabatier040.ib:514
$ActionExecOnlyWhenPreviousIsSuspended on
# si suspend, alors on envoie les logs sur le serveur de secours (failover)
& @@sabatier041.ib:514
& @@sabatier042.ib:514
& @@sabatier042.ib:514
# reset pour le prochain selecteur
$ActionExecOnlyWhenPreviousIsSuspended off
```

Dans le cas des nœuds de calcul, si on ne souhaite envoyer que les logs indiquant un problème, on peut remplacer ***.*** par ***.warning***.

9 Synchronisation temporelle du cluster

Afin de synchroniser temporellement notre cluster, nous avons décider d'opter pour une architecture NTP résiliente.

Nous avons décidé que chacun des 4 serveurs NTP (hébergés sur 4 noeuds de service) se synchronisent en mode client-serveur sur les deux serveurs NTP de l'Université en utilisant la connexion au backbone des noeuds de service. En cas de panne des serveurs NTP de l'Université ou de coupure du backbone, les quatre serveurs basculent en mode symétrique pour maintenir une date et une heure précises entre eux. Les autres noeuds du cluster vont alors se synchroniser sur ces 4 noeuds.

Il n'est à priori pas nécessaire de modifier la configuration des serveurs NTP de l'université, ceux-ci ayant des IPs publiques, on peut supposer que n'importe qui est autorisé à s'y connecter en mode client-serveur pour récupérer le temps. Les 4 noeuds de service choisis pour héberger des serveurs NTP sont sabatier[44-47].

Voici les fichiers de configuration :

Listing 4: Configuration des serveurs NTP (4 noeuds de service)

```
# client-server synchronization
# using the burst option on public servers is considered rude (generates more traffic) but ↩
    in this case it's fine because this node acts as a server too so it needs more precise ↩
    time control
# https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/↩
    system_administrators_guide/ch-configuring_ntp_using_ntpd#↩
    s2_Configuring_the_Burst_Option
pool ossau1.univ-tlse3.fr iburst burst
pool ossau2.univ-tlse3.fr iburst burst

# symetric synchronization
peer sabatier044.admin
peer sabatier045.admin
peer sabatier046.admin
peer sabatier047.admin

# default access control
restrict default nomodify notrap nopeer noquery
```

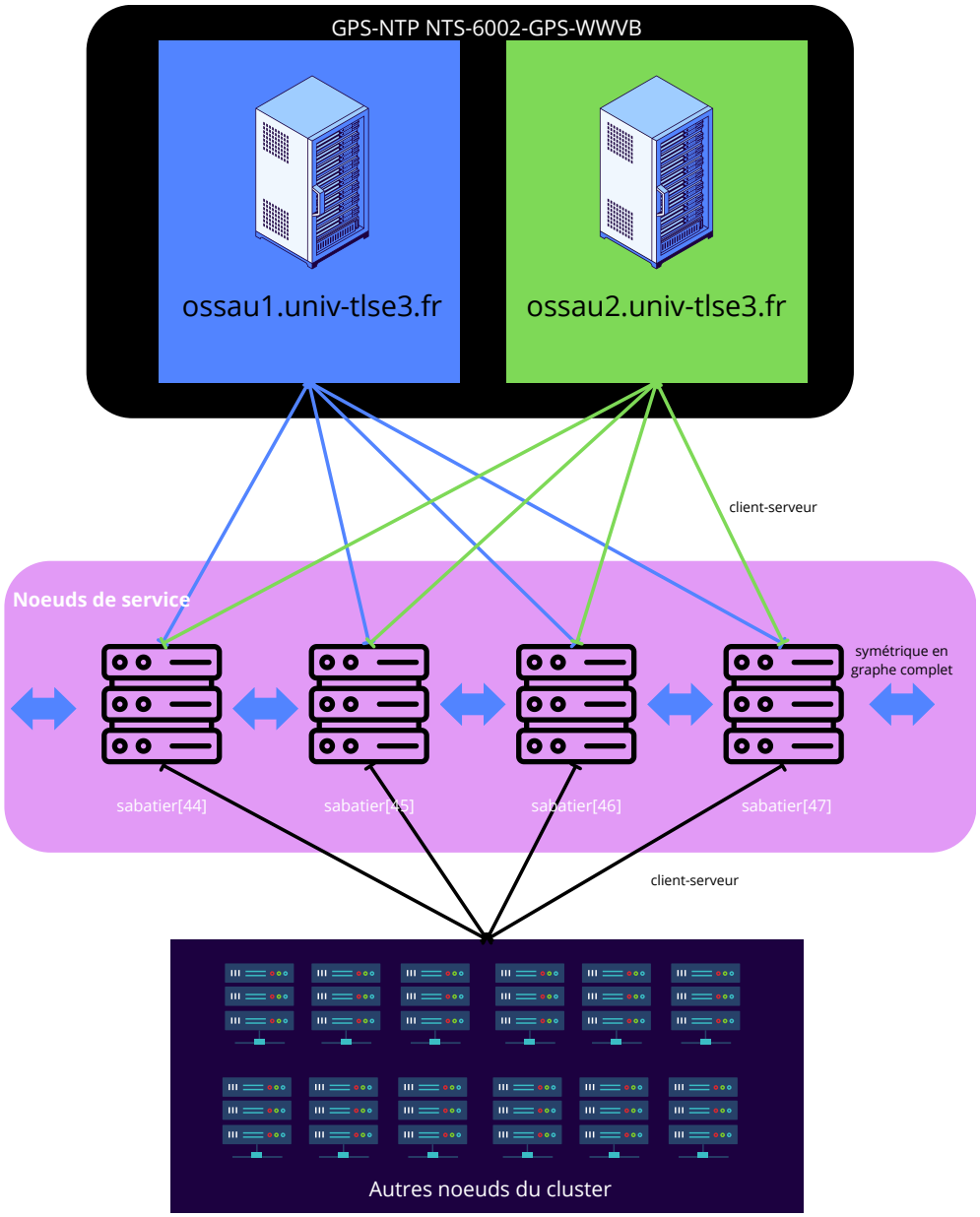
où sur chacun des 4 noeuds de service hébergeant un serveur NTP, il faudrait commenter la ligne peer correspondant au noeud lui-même. Quant à la configuration client :

Listing 5: Configuration des clients NTP (tous les autres noeuds)

```
# client-server synchronization
# not using the burst option to avoid overwhelming service nodes with 6x more requests, as ↩
    there are more than 400 nodes
# https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/↩
    system_administrators_guide/ch-configuring_ntp_using_ntpd#↩
    s2_Configuring_the_Burst_Option
pool sabatier044.admin iburst
pool sabatier045.admin iburst
pool sabatier046.admin iburst
pool sabatier047.admin iburst

# default access control
```

```
restrict default nomodify notrap nopeer noquery
```

Le mot-clé **iburst** permet d'envoyer une rafale de requêtes si le serveur ne répond pas, ce qui permet de se synchroniser dessus au plus vite, et le mot-clé **burst** permet d'envoyer une rafale de requêtes lorsque le serveur répond, pour avoir une estimation plus précise du décalage en temps due aux transferts de paquets. L'usage de **burst** en tant que client auprès de serveurs publics est très mal vu, car cela multiplie les requêtes pour avoir un temps à peine plus précis, cependant l'usage qui en est fait ici sert à améliorer la précision du temps des noeuds de service, car il n'y en a que 4 et ils servent eux-mêmes de serveurs NTP, l'usage de **burst** est donc ici tout à fait justifié. Les autres noeuds (clients NTP) du cluster ne s'en servent pas, car il y a plus de 400 noeuds, et multiplier les requêtes envoyées par tous ces noeuds vers seulement 4 noeuds de service ne serait pas raisonnable. En supposant que la plupart du temps, les serveurs NTP (internes et externes au cluster) sont contactables, il semble raisonnable d'utiliser l'option **iburst** dans les deux cas.



10 Architecture LDAP

Les annuaires LDAP hébergés sur les 3 nœuds de service doivent être des répliques identiques de l'annuaire présent sur le service LDAP de l'université (ou une copie partielle si toutes les informations ne sont pas nécessaires). Pour cela, l'idée est que l'un des 3 nœuds de service mettra régulièrement à jour son contenu en envoyant des requêtes au service LDAP de l'université, et les deux autres nœuds de service synchronisent leur annuaire en utilisant ce premier nœud de service. Idéalement, afin d'éviter un SPOF, si le nœud de service désigné n'a plus accès au service LDAP de l'université, un autre doit prendre ce rôle (failover).

Les 3 nœuds de service utilisés sont sabatier{40,41,48}, c'est-à-dire 3 nœuds sur des switchs différents à la fois sur le réseau d'administration et le réseau IB, et le nœud désigné pour échanger avec le service de l'université est sabatier48.

Les nœuds du cluster ayant besoin d'utiliser le service LDAP n'enverront des requêtes qu'à ces trois nœuds de service. La configuration du service LDAP de l'université doit être modifiée pour permettre aux 3 nœuds de service de recopier le contenu ou une partie de l'annuaire.

Fichiers de configuration :

Listing 6: Configuration des clients LDAP

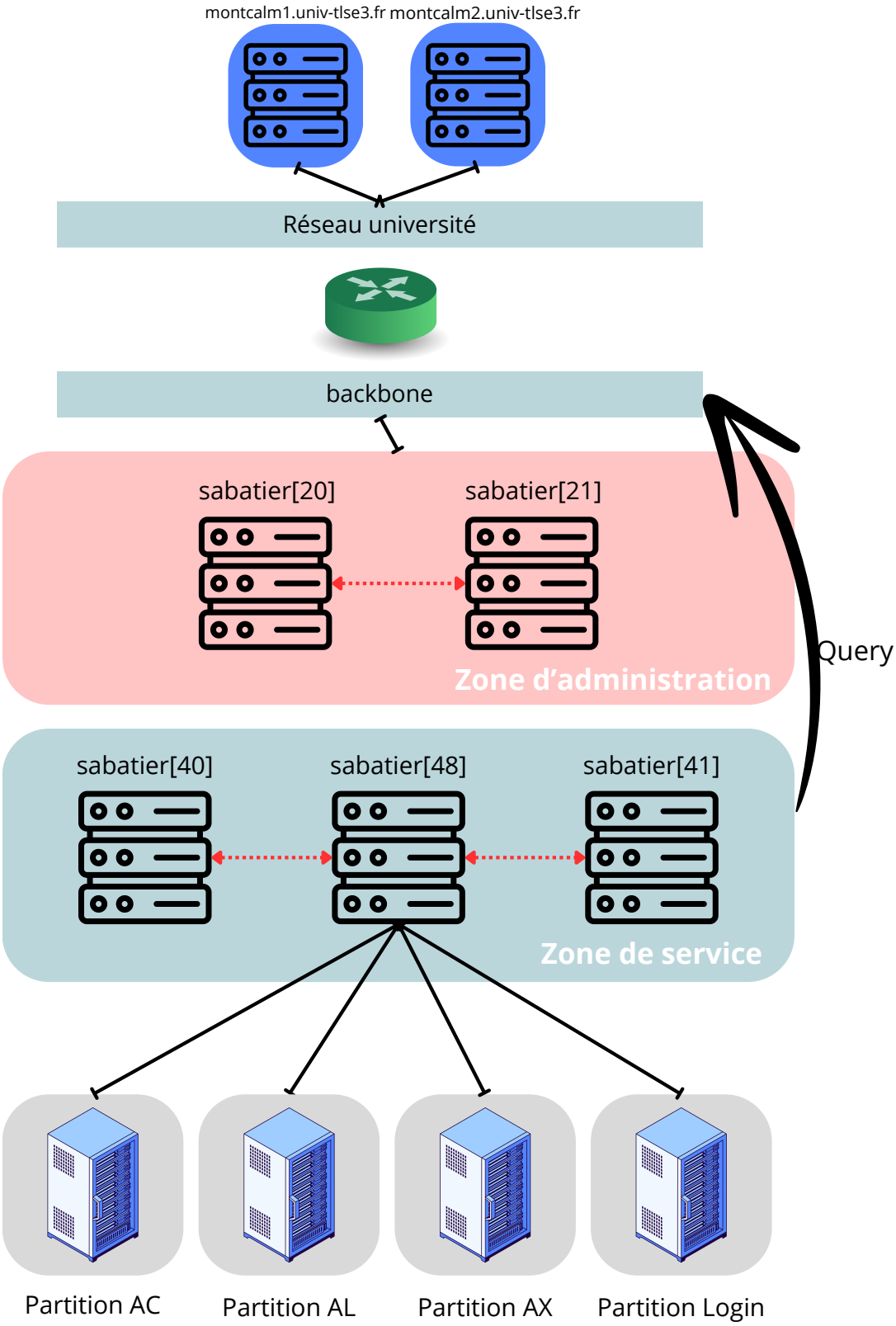
```
# See ldap.conf(5) for details
# This file should be world readable but not world writable.

BASE dc=sabatier,dc=univ-tlse3, dc=fr

# serveur ldap service maitre
URI ldap://sabatier048.admin

# serveur esclave du service maitre
URI ldap://sabatier040.admin
URI ldap://sabatier041.admin

#SIZELIMIT 12
#TIMELIMIT 15
#DEREF  never

TLS_CACERT /etc/openldap/certs
SASL_MECH gssapi
```

Listing 7: Configuration des serveurs LDAP du cluster

```
# See ldap.conf(5) for details
include /etc/openldap/schema/core.schema
include /etc/openldap/schema/cosine.schema

argsfile /var/run/openldap/slapd.args
pidfile /var/run/openldap/slapd.pid

moduleload syncprov.la

access to *
    # pour que les autres serveurs master puissent ecrire
    # car on fait du multi-master
```

```
    by dn.children="ou=service,dc=sabatier,dc=univ-tlse3,dc=fr" write
    by * read

########################################################################

# Multimaster replication

ServerID 1 "ldap://sabatier040.admin"
ServerID 2 "ldap://sabatier041.admin"
overlay  syncprov

# multimaster enabled
syncrepl rid=001
       provider=ldap://sabatier048.admin
       type=refreshAndPersist
       interval=00:00:00:10
       retry="5 10 60 +"
       searchbase="dc=univ-tlse3,dc=fr"
       filter="(objectClass=*)"
       scope=sub

MirrorMode on

########################################################################

TLSCACertificatePath /etc/openldap/certs
TLSCertificateFile /etc/openldap/certs/domain.pem
TLSCertificateKeyFile /etc/openldap/certs/domain.pem

database bdb
suffix "dc=univ-tlse3,dc=fr"
directory /var/lib/ldap

rootdn "cn=admin,dc=univ-tlse3,dc=fr"
rootpw "root_is_the_best_password"

database monitor
```

Illustration de l'architecture LDAP :



11 Architecture DNS

11.1 Convention d'adressage

Avant d'attribuer des noms de domaine à associer aux adresses IP des interfaces des noeuds du cluster, il faut avant tout attribuer des adresses IP à ces interfaces. On établit pour cela une convention d'adressage. Ainsi, l'ajout de noeuds au cluster ne devrait pas poser de problème : il suffit de suivre la convention pour attribuer des adresses IP à toutes leurs interfaces. On supposera ici que si un noeud a de multiples liens physiques vers un même réseau, il est agrégé en un unique lien logique (et par extension, chaque noeud aura une adresse unique et un nom de domaine unique sur chaque réseau).

Nous attribuons dans un premier temps des plages réseau :

- le réseau d'administration est 172.16.0.0/16
- le réseau InfiniBand est 172.17.0.0/16
- le réseau de contrôle électrique est 172.18.0.0/16
- la portion du réseau backbone attribuée au cluster est 172.19.0.0/16 (l'université peut en décider autrement si celle-ci est déjà attribuée, mais du moment qu'elle est de la forme 172.X.0.0/16 il n'y aura pas de problème)

Pour chaque noeud maître, un port 1 Gb est relié directement au port BMC de l'autre, ces deux ports ne sont donc pas sur le réseau de contrôle électrique, on pourra leur attribuer une plage IP telle que 10.0.0.0/31 pour le noeud maître 1 et la BMC du noeud maître 2, et 10.0.1.0/31 pour le noeud maître 2 et la BMC du noeud maître 1 (utiliser le /31 est assez courant pour des routeurs connectés deux à deux, qui n'utilisent alors pas de switch et n'ont besoin que de 2 adresses). Dans le cas d'un noeud maître, on pourra donner un alias à la BMC de l'autre noeud maître dans /etc/hosts, vu que ce sont les 2 seuls noeuds ayant besoin de connaître ces interfaces.

L'adresse attribuée à un noeud qui n'est pas un noeud de calcul est la suivante :
172.X.Y.Z avec X qui dépend du réseau, et Y et Z qui correspondent au numéro attribué au noeud d'après la configuration clustershell, où Y=100+numéro/100 et X=numéro%100.
Exemples :

- sur le réseau d'administration, pour le noeud maître 1, on a X=16 car c'est le réseau d'administration, Y=100 et Z=20 car c'est le noeud sabatier20. On a alors 172.16.100.20 pour cette interface.
- pour le 155e noeud de la partition AC (sabatier454), son adresse sur le réseau InfiniBand est 172.17.104.54

pour s'adresser à une machine au sein du cluster, l'usage des noms d'hôte ne suffit pas : il faut savoir sur quel réseau envoyer des requêtes. On utilisera alors le serveur DNS qu'on essaie de mettre en place pour permettre aux machines du cluster de communiquer entre elles en utilisant des noms de domaine contenant deux informations : le nom de la machine, et le réseau utilisé.

11.2 Configuration DNS

Le serveur DNS du cluster (hébergé sur les noeuds de service sabatier[43-44] pour la redondance) établit les correspondances entre des noms de domaine et des adresses IP pour les services de l'université (il a alors recours au DNS de l'université) et pour les interfaces logiques des noeuds du cluster (configuration statique). Il n'est accessible que via le réseau d'administration, c'est-à-dire depuis l'intérieur du cluster.

Il y a également quelques changements à faire sur la configuration DNS de l'université pour qu'il attribue des noms de domaine aux noeuds du cluster présents sur le backbone : les noeuds de login pour les utilisateurs, les noeuds maîtres pour les administrateurs ainsi que les noeuds de service et les routeurs. Les noms donnés sur le backbone seront volontairement plus explicites.

La convention de nommage des noeuds du cluster est la suivante, quelque soit X :

- le nom sabatierX.admin est associé à l'adresse de la machine sabatierX sur le réseau d'administration
- le nom sabatierX.ib est associé à l'adresse de la machine sabatierX sur le réseau InfiniBand
- le nom hpcX.sabatier.univ-tlse3.fr est associé à l'adresse du noeud de login numéro X en comptant à partir de 0 (sabatierX) sur le backbone
- le nom lustreX.sabatier.univ-tlse3.fr est associé à l'adresse du routeur numéro X en comptant à partir de 0 (sabatier060+X) sur le backbone
- le nom hpcadminX.sabatier.univ-tlse3.fr est associé à l'adresse du noeud maître numéro X en comptant à partir de 0 (sabatier020+X) sur le backbone
- le nom hpcsrvX.sabatier.univ-tlse3.fr est associé à l'adresse du noeud de service numéro X en comptant à partir de 0 (sabatier040+X) sur le backbone

Les noms de domaines internes au réseau du cluster n'ont pas besoin de se terminer par univ-tlse3.fr, car c'est un domaine au sein d'un réseau privé, on leur a donc choisi des noms courts qui donnent suffisamment d'informations.

Les autres noms de domaine (externes au cluster) seront récupérés sur les serveurs DNS de l'université. Cependant, même si cela permet à tous les noeuds du cluster de résoudre un nom de domaine venant d'Internet, une requête vers l'adresse IP associée n'aboutira pas forcément, car seuls les noeuds maîtres, accessibles uniquement aux administrateurs, auront le droit d'échanger avec Internet sans restriction.

Le DNS interne du cluster doit savoir résoudre les noms de la forme sabatierX.admin,ib,ctrl et les adresses de la forme 172.[16-18].X.Y, et transmettre au serveur DNS de l'Université les requêtes qu'il ne peut pas satisfaire (fonctionnement récursif). Les deux noeuds de service choisis pour le DNS sont indépendants et ne fonctionnent pas en maître/esclave l'un de l'autre, l'objectif étant la redondance, on évite l'interdépendance.

On commence alors par établir une configuration pour les clients DNS (tous les noeuds du cluster) dans /etc/resolv.conf :

Listing 8: resolv.conf, client

```
# univ-tlse3.fr contains University services subdomains,
search admin ib ctrl univ-tlse3.fr hpc.univ-tlse3.fr
# sabatier43, DNS server
```

```
nameserver 172.16.100.43
# sabatier44, DNS server
nameserver 172.16.100.44
```

et dans /etc/nsswitch.conf :

Listing 9: nsswitch.conf, client

```
hosts: files dns
```

Quant à la configuration des 2 noeuds de service, on retrouve le fichier de configuration DNS /etc/named.conf sur le noeud sabatier43 :

Listing 10: named.conf, serveur

```
//
// named.conf
//
// Provided by Red Hat bind package to configure the ISC BIND named(8) DNS
// server as a caching only nameserver (as a localhost DNS resolver only).
//
// See /usr/share/doc/bind*/sample/ for example named configuration files.
//
// See the BIND Administrator's Reference Manual (ARM) for details about the
// configuration located in /usr/share/doc/bind-{version}/Bv9ARM.html

// nodes can only contact the DNS through the admin network
acl cluster {
    172.16.0.0/16;
};

acl univdns {
    // marbore1
    172.4.24.13;
    // marbore2
    172.4.24.14;
}

options {
    // interface we're listening on
        listen-on port 53 { 127.0.0.1; 172.16.100.43; };
        listen-on-v6 port 53 { ::1; };
        directory  "/var/named";
        dump-file  "/var/named/data/cache_dump.db";
        statistics-file "/var/named/data/named_stats.txt";
        memstatistics-file "/var/named/data/named_mem_stats.txt";
        recursing-file "/var/named/data/named.recursing";
        secroots-file "/var/named/data/named.secroots";
        allow-query { localhost; cluster; };
    forward only;
    forwarer { univdns; };

        /*
         - If you are building an AUTHORITATIVE DNS server, do NOT enable recursion.
         - If you are building a RECURSIVE (caching) DNS server, you need to enable
           recursion.
```

```
        - If your recursive DNS server has a public IP address, you MUST enable access
          control to limit queries to your legitimate users. Failing to do so will
          cause your server to become part of large scale DNS amplification
          attacks. Implementing BCP38 within your network would greatly
          reduce such attack surface
       */
       recursion yes;

       dnssec-enable yes;
       dnssec-validation yes;

       /* Path to ISC DLV key */
       bindkeys-file "/etc/named.root.key";

       managed-keys-directory "/var/named/dynamic";

       pid-file "/run/named/named.pid";
       session-keyfile "/run/named/session.key";
};

logging {
       channel logfile {
           file "/var/log/named.log";
           severity dynamic ;
           print-category yes ;
           print-time yes ;
       };
       category queries { logfile ; } ;
       category config { logfile ; } ;
       category xfer-out { logfile ; } ;
       category xfer-in { logfile ; } ;
       category network { logfile ; } ;
       category default { logfile ; } ;
};

zone "." IN {
       type hint;
       file "named.ca";
};

zone "admin" IN {
       type master;
       file "named.admin";
};

zone "ib" IN {
       type master;
       file "named.ib";
};

zone "ctrl" IN {
       type master;
       file "named.ctrl";
};
```

```
zone "16.172.in-addr.arpa" IN {
       type master;
       file "named.16.172";
};

zone "17.172.in-addr.arpa" IN {
       type master;
       file "named.17.172";
};

zone "18.172.in-addr.arpa" IN {
       type master;
       file "named.18.172";
};

include "/etc/named.rfc1912.zones";
include "/etc/named.root.key";
```

où les fichiers /var/named/named.admin,ctrl,ib et /var/named/named.[16-18].172 contiennent les associations entre noms de domaine et adresses IP et inversement, selon la convention d'adressage établie précédemment. Le noeud sabatier44 contient les mêmes fichiers, mis à part que toutes les occurrences de sabatier43 et 172.16.100.43 sont remplacées respectivement par sabatier44 et 172.16.100.44.

12 Configuration du contrôleur SLURM

12.1 Choix des serveurs et configuration de slurm

Afin d'utiliser slurm, il est nécessaire que celui-ci ainsi que ses dépendances soit installés sur les nœuds de calculs, c'est-à-dire slurmd, munge.

Pour les nœuds de services, s'occupant de slurm, il sera nécessaire d'installer, slurmd, slurmctld, munge. Afin d'avoir une comptabilité des ressources consommées ainsi qu'une persistance des données, il sera nécessaire d'utiliser slurmdbd. Nous utiliserons MariaDB qui est une base de données MySQL.

Dans notre configuration Slurm, nous utiliserons le "backfill" pour l'ordonnancement des jobs, cela permettra à un job moins prioritaire de s'exécuter tant qu'il ne modifie pas le démarrage d'autre job. De plus, pour le choix des nœuds, il nous est demandé de permettre aux utilisateurs de choisir leurs nœuds. Nous utiliserons donc la sélection des ressources "con_tres".

Nous choisirons les 2 nœuds de services sabatier 48 et 49. Sabatier 49 sera le SlurmctldHost, c'est-à-dire le nœud maître de la configuration Slurm ou celui-ci contrôle les autres daemons slurm, l'autre nœud sera le Backup en cas de problème avec le nœud maître. Nous choisissons ces nœuds, car ce sont ceux qui sont le moins utilisé par les autres services (rsyslog, ntp, etc.).

Listing 11: Configuration slurm.conf

```
ClusterName=sabatier
SlurmctldHost=sabatier049.ib
SlurmctldHost=sabatier048.ib
AuthType=auth/munge
#
MaxJobsPerUser = 100
MaxSubmitJobsPerUser = 100
MaxJobCount=10000
#MaxStepCount=40000
#MaxTasksPerNode=512
#MpiDefault=
#MpiParams=ports=#-#
#PluginDir=
#PlugStackConfig=
#PrivateData=jobs
ProctrackType=proctrack/cgroup
#Prolog=
#PrologFlags=
#PrologSlurmctld=
#PropagatePrioProcess=0
#PropagateResourceLimits=
#PropagateResourceLimitsExcept=
#RebootProgram=
ReturnToService=1
SlurmctldPidFile=/var/run/slurmctld.pid
SlurmctldPort=6817
SlurmdPidFile=/var/run/slurmd.pid
SlurmdPort=6818
SlurmdSpoolDir=/var/spool/slurmd
SlurmUser=root
```

```
StateSaveLocation=/var/spool/slurmctld
TaskPlugin=task/affinity,task/cgroup
#
# TIMERS
#BatchStartTimeout=10
#CompleteWait=0
#EpilogMsgTime=2000
#GetEnvTimeout=2
#HealthCheckInterval=0
#HealthCheckProgram=
InactiveLimit=0
KillWait=30
#MessageTimeout=10
#ResvOverRun=0
MinJobAge=300
#OverTimeLimit=0
SlurmctldTimeout=120
SlurmdTimeout=300
#UnkillableStepTimeout=60
#VSizeFactor=0
Waittime=0
#
#
# SCHEDULING
#DefMemPerCPU=0
#MaxMemPerCPU=0
#SchedulerTimeSlice=30
SchedulerType=sched/backfill
SelectType=select/cons_tres
SelectTypeParameters=CR_Core
#
#
# JOB PRIORITY
#PriorityFlags=
#PriorityDecayHalfLife=
#PriorityCalcPeriod=
#PriorityFavorSmall=
#PriorityMaxAge=
#PriorityUsageResetPeriod=
#PriorityWeightAge=
PriorityType=priority/multifactor
PriorityWeightFairShare=100000
PriorityWeightJobSize=0
PriorityWeightPartition=0
PriorityWeightQOS=1000000
#
#
# LOGGING AND ACCOUNTING
#AccountingStorageEnforce=0
AccountingStorageEnforce=qos,limits
AccountingStorageHost=sabatier049.ib
#AccountingStoragePass=
#AccountingStoragePort=
AccountingStorageType=accounting_storage/slurmdbd
```

```
#AccountingStorageUser=
#AccountingStoreFlags=
#JobCompHost=
#JobCompLoc=
#JobCompParams=
#JobCompPass=
#JobCompPort=
JobCompType=jobcomp/mysql
#JobCompUser=
#JobContainerType=
JobAcctGatherFrequency=30
#JobAcctGatherType=
SlurmctldDebug=info
SlurmctldLogFile=/var/log/slurmctld.log
SlurmdDebug=info
SlurmdLogFile=/var/log/slurmd.log
#SlurmSchedLogFile=
#SlurmSchedLogLevel=
#DebugFlags=
#
#
# POWER SAVE SUPPORT FOR IDLE NODES (optional)
#SuspendProgram=
#ResumeProgram=
SuspendTimeout=300
ResumeTimeout=300
ResumeRate=0
#SuspendExcNodes=
SuspendExcParts=debug
SuspendRate=0
SuspendTime=600
#
#
# COMPUTE NODES
NodeName=sabatier[060-069] State=UNKNOWN # AX
NodeName=sabatier[200-249] State=UNKNOWN # AL
NodeName=sabatier[300-601] State=UNKNOWN #AC
PartitionName=AX Nodes=sabatier[060-069] Default=YES MaxTime=INFINITE State=UP
PartitionName=AL Nodes=sabatier[200-249] Default=YES MaxTime=INFINITE State=UP
PartitionName=AC Nodes=sabatier[300-601] Default=YES MaxTime=INFINITE State=UP
```

Étant donnée que ce centre sera utilisé par une université, il est possible que par moment celui-ci soit très peu utilisé (exemple : vacance scolaire). Afin de prévenir une consommation excessive des noeuds alors que ceux-ci sont en "idle", nous indiquons également dans la configuration slurm que ceux-ci peuvent se suspendre si aucun job n'a été exécuter pendant plus de 10 minutes.

Listing 12: Configuration slurmdbd.conf

```
AuthType=auth/munge
DbdHost=localhost
SlurmUser=root
DebugLevel=4
LogFile=/var/log/slurm/slurmdbd.log
PidFile=/var/run/slurmdbd.pid
StorageType=accounting_storage/mysql
```

```
StoragePass=<password>
StorageUser=root
```

Pour créer la base de données MariaDB, nous entrons ces instructions dans l'invite de commande après avoir fait sudo mysql.

Listing 13: Création de la base de donnée mariadb

```
grant all on slurm_acct_db.* TO 'root'@'localhost' identified by '<password>'
with grant option;
create database slurm_acct_db;
quit;
```

De plus, il est recommandé de rajouter une configuration MySQL afin de gérer les fichiers de taille importante :

Listing 14: Configuration innodb.conf

```
[mysqld]
innodb_buffer_pool_size=32768M
innodb_log_file_size=64M
innodb_lock_wait_timeout=900
```

Il est indiqué sur le site de slurm qu'il est possible d'automatiser la génération de la configuration pour la topologie Infiniband. Étant donnée que nous utilisons de l'Infiniband, nous modifierons topology.conf en utilisant celui-ci (ici).

12.2 Restriction des ressources

Pour qu'il y ait restriction des ressources en fonction des projets, il sera nécessaire de créer des partitions, nous en créerons 1 par projet. Ensuite, pour configurer les restrictions, on créera un utilisateur parent qui contiendra les restrictions que l'on souhaite appliquer aux utilisateurs. Chaque utilisateur sera rattaché au parent de son projet.

Nous avons décidé de limiter le nombre de jobs par utilisateur à 10 (MaxJobsPerUser=10), de plus, chaque utilisateur ne pourra lancer que 100 jobs et en avoir 100 en attente (situé dans slurm.conf). La limitation des ressources par défaut (mémoire, cpu, etc.) se fait en créant un utilisateur avec sacctmgr et dépendra des instructions données par l'université.

Afin d'offrir la possibilité aux utilisateurs des projets 1 ou 2 d'avoir accès au debug de leurs jobs avec une priorité élevé, nous allons mettre en place une QOS nommée "debug". Pour faire cela, nous utiliserons sacctmgr.

Listing 15: Création de la QOS debug

```
sacctmgr -i create QOS name=debug priority=10 maxjobs=2 maxwall=10:00
```

Nous avons mis la limite de job par utilisateur à 10 dans le fichier de configuration slurm, nous indiquerons donc que les utilisateurs ne pourront débuger que 2 jobs, et que ceux-ci auront une durée maximale de 10 minutes.

Il nous reste à créer les utilisateurs parents, ceux-ci seront directement reliés au projets et les utilisateurs seront rattachés au parents.

Listing 16: Création des utilisateurs parents

```
sacctmgr create account name=projet0 faishare=15 partition=AC,AL # on exclue la partie AX
sacctmgr create account name=projet1 faishare=20
sacctmgr create account name=projet2 faishare=40
sacctmgr create account name=projet3 faishare=25
```

Nous attribuons respectivement aux 4 projets 15 %, 20 %,40 %, 25 % d'utilisation de la machine. Il ne reste plus qu'a crée les utilisateurs et les associer aux parents.

Dans l'énoncé, les ugp ne sont pas réparti équitablement, il est marqué pour le projet 0 que ceux-ci vont de 0 à 2 inclus (donc il y en a 3) tandis que pour le projet 3 il n'y a que 2 ugp (8 et 9), nous ne savons pas comment sont répartis les ugp du projet 1 et 2. Nous les redéfinissons ainsi afin qu'il y ait 2 ugp par groupe.

- projet 0 : ugp[0-1]
- projet 1 : ugp[2-4]
- projet 2 : ugp [5-7]
- projet 3 : upg [8-9]

Pour le projet 0, nous créons un utilisateur de cette manière :

Listing 17: Création de l'utilisateur 1 associé au projet 0

```
sacctmgr create account name=user1ugp0projet0 parent=project0 qos=normal
```

Ainsi qu'un utilisateur pour le projet 1 et 2, ils doivent donc pouvoir debugger rapidement, en plus de pouvoir lancer des jobs.

Listing 18: Création des utilisateurs

```
sacctmgr create account name=user2ugp2projet1 parent=project1 qos=debug,normal
sacctmgr create account name=user3ugp5projet2 parent=project2 qos=debug,normal
```

Enfin, nous créeons un utilisateur pour le projet 3.

Listing 19: Création des utilisateurs

```
sacctmgr create account name=user4ugp8projet3 parent=project3 qos=normal
```

12.3 Vérification de la politique mise en place

Pour vérifier la politique (QOS) mise en place, nous effectuerons la commande suivante :

Listing 20: Vérification QOS

```
sacctmgr show qos
```

Pour voir si les partitions ainsi que les noeuds associés ont bien été crées, nous pouvons utiliser la commande :

Listing 21: Vérification partition + noeud

```
sinfo
```

Pour vérifier si les fairshares ont bien été appliqués pour les utilisateurs parents :

Listing 22: Vérification fairshare

```
sacctmgr -nP list associations user=projet0 format=fairshare
```

Et pour voir les informations mises dans le fichier de configuration slurm :

Listing 23: Vérification slurm.conf

```
sacctmgr -nP show clusters format=cluster
```

13 Configuration de la surveillance

13.1 Utilisation des sondes et mise en place d'une sonde

La société AzkharHPC nous donne des sondes afin de vérifier les critères de disponibilités. On suppose que ces sondes sont des sondes classiques avec un message court sur stdout. Il y a très peu d'informations sur les sondes, on suppose qu'elles sont bien faites et respecte les conventions vue en cours. C'est-à-dire :

- -V version (–version)
- -h help (–help)
- -t timeout (–timeout)
- -w warning threshold (–warning)
- -c critical threshold (–critical)
- -H hostname (–hostname)
- -v verbose (–verbose)

De plus, on suppose également que parmi les scripts, ceux-ci fonctionnent ainsi :

- verif-noeud, donne le pourcentage sur une liste de noeud up (exemple : si on fait 'verif-noeud -H a,b,c,d' et que seul le noeud d est up, il rendra 25 (pour 25%).
- verif-lustre-routeur, donne le débit.
- verif-nodeset-ssh, donne les pourcentages, comme verif-noeud.

Listing 24: Utilisation des sondes pour vérifier les critères de disponibilité

```
# 1) Au moins un noeud de login doit etre operationnel pour chaque communaute
verif-nodeset-ssh -H [liste-noeud-login] -w 1 -c 0

# 2) Un debit minimal de 32 GB/s vers le systeme de stockage Lustre doit etre assure,
verif-lustre-routeur -H [liste-noeud-lustre] -w 35 -c 31

# 3) 90% des noeuds de la partition AC doivent etre disponibles
verif-noeud -H [liste-noeud-AC] -w 95 -c 89

# 4) 80% des noeuds de la partition AL,
verif-noeud -H [liste-noeud-AL] -w 85 -c 79

# 5) 60 % des noeuds de la partition AX.
verif-noeud -H [liste-noeud-AX] -w 65 -c 59
```

Lors du TP de surveillance, il était demandé d'écrire une sonde en bash. Cependant, nous pouvons voir que lors de celui-ci nous avons également installer des plugins. Dont linux-ssh, qui lui a des checks écrit en Python (code disponible ici). Nous ferons donc l'hypothèse que nous pouvons écrire nos check en Python 2, qui est le Python utilisé pour l'écriture des checks linux-ssh.

Afin de surveiller les seuils de disponiblités pour une partition slurm, nous proposons le script suivant :

Listing 25: Sonde de vérification de la disponibilité d'une partition slurm

```
# inspired by: /usr/lib64/nagios/plugins/ in the slurm vm
# Lapu Matthias , Louis Auffret , ENSIIE

import os
import sys
import optparse
import subprocess
try:
    import paramiko
except ImportError:
    print("ERROR : this plugin needs the python-paramiko module. Please install it"↩
        )
    sys.exit(2)

# Load plugin utils
my_dir = os.path.dirname(__file__)
sys.path.insert(0, my_dir)

try:
    import schecks
except ImportError:
    print("ERROR : this plugin needs the local schecks.py lib. Please install it")
    sys.exit(2)

VERSION = "0.1"

def execute_slurm_check(client, command):
    stdin, stdout, stderr = client.exec_command(command)
    output = stdout.read().decode().strip()
    client.close()

    return output

# output to process :
# [matthias.lapu@hpc01 ~]$ sinfo
# PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
# inter        up   infinite      2   idle hpc[02-03]
# calcul*      up   infinite     10   idle hpc[04-13]

def check_slurm_disponibility_partition(output, partition):
    # += should'nt be necessary because there should only be
    # one line with the partition name but we never know
    node_count = 0
    lines = output.split('\n')
    for line in lines:
        if partition in line:
            # we need to count the number of nodes in the partition
            node_count += int(line.split()[3])
    # now we need to have the number of running jobs
    available_nodes = 0
    for line in lines:
        if partition in line:
            if "idle" in line:
                available_nodes += int(line.split()[3])
    return "OK : %d nodes available out of %d" % (available_nodes, node_count)

parser = optparse.OptionParser(
```

```
    "%prog [options]", version="%prog " + VERSION)
parser.add_option('-H', '--hostname',
                  dest="hostname", help='Hostname to connect to')
########Added ########
parser.add_option('--partition',
                  dest="partition", help='Partition name to check the availability'↩
                     )
######################
parser.add_option('-p', '--port',
    dest="port", type="int", default=22,
    help='SSH port to connect to. Default : 22')
parser.add_option('-i', '--ssh-key',
                  dest="ssh_key_file", help='SSH key file to use. By default will ↩
                     take ~/.ssh/id_rsa.')
parser.add_option('-u', '--user',
                  dest="user", help='remote use to use. By default shinken.')
parser.add_option('-P', '--passphrase',
                  dest="passphrase", help='SSH key passphrase. By default will use ↩
                     void')
parser.add_option('-r', '--check_path',
                  dest="check_path", help='Path of the remote perfdata check to ↩
                     execute')


if __name__ == '__main__':
    opts, args = parser.parse_args()
if args:
    parser.error("Does not accept any argument.")

if not opts.hostname:
    print("Error : hostname parameter (-H) is mandatory")
    sys.exit(2)

# we must check the command is used properly
# we need the partition, warning and critical thresholds
partition = opts.partition
if not partition:
    print("Error : partition parameter (--partition) is mandatory")
    sys.exit(2)
warning = opts.warning
if not warning:
    print("Error : warning parameter (-w) is mandatory")
    sys.exit(2)
critical = opts.critical
if not critical:
    print("Error : critical parameter (-c) is mandatory")
    sys.exit(2)

client= schecks.connect_ssh(opts.hostname, opts.port, opts.user, opts.ssh_key_file,↩
    opts.passphrase)
output = execute_slurm_check(client, partition)
result = check_slurm_disponibility_partition(output, opts.threshold)
print(result)
sys.exit(0)
```

Cette sonde indiquera donc le niveau de disponibilité d'une partition par le biais de slurm et pourra alors être utilisé ainsi :

Listing 26: Utilisation de notre sonde

```
# par exemple, en ayant cela :
# PARTITION AVAIL  TIMELIMIT  NODES  STATE      NODELIST
```

```
# inter        up   infinite      1   allocated hpc[03]
# inter        up   infinite      1   idle      hpc[02]
# calcul*      up   infinite     10   idle      hpc[04-13]



# sonde sur la partition inter : si <25% warning, si <10% critical
check_slurm_disponibility_partition --partition="inter" -w 25 -c 10
```

13.2 Analyse des noeuds

Afin qu'un nœud de login soit interactif et confortable, il est nécessaire de surveiller différents aspects.

Premièrement, nous devons surveiller sa charge, en effet, un nœud de login doit avoir suffisamment de ressources matérielles telles que la RAM, le CPU et le stockage pour gérer efficacement les charges de travail des utilisateurs. Nous pensons que le stockage est un point important pour les utilisateurs, il est donc important de surveiller son taux de saturation. C'est en vérifiant cela que celui-ci sera aperçu comme interactif du point de vue des utilisateurs.

Ensuite, la disponibilité du réseau semble être un autre point important, une latence trop élevée ne sera pas bien vu du point de vue utilisateur et cette latence nuira à l'expérience, car le nœud ne sera pas "interactif". Il faudra donc également analyser la latence des nœuds. Cette latence peut venir de différents facteurs que ce soit un problème de certificat LDAP (comme expériencé à l'ENSIIE au début de l'année 2024) , une saturation du réseau, etc. Cette latence pourrait nuire à la qualité de service et faire perdre des clients.

Enfin, l'accessibilité des nœuds est également un aspect important, sinon les utilisateurs ne pourront s'y connecter.

Au niveau d'un centre de calcul, les nœuds d'administration et de services sont des services critiques à surveiller, en effet, ceux-ci regroupent les servies les plus importants au bon fonctionnement du centre. Nous pensons également qu'il est important de surveiller le stockage Lustre, celui-ci possède un nombre conséquent de données, et les pertes de ceux-ci peuvent être handicapantes pour les utilisateurs.

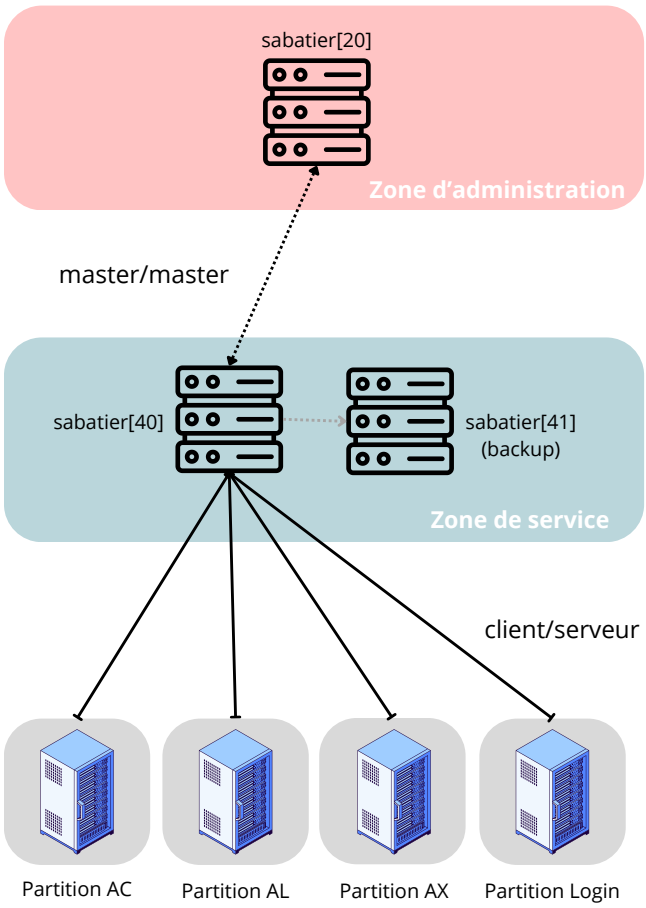
Afin d'effectuer une analyse plus fine des performances, il faudrait mettre en place des checks avec plus d'options et étant plus précis. Si nous souhaitons utiliser un autre outil, nous utiliserons alors Prometheus, qui est un système de monitoring. En effet, il serait possible d'avoir de manière scalable et automatique, de nombreuses données à intervalles régulier. De plus, celui-ci nous permettrait de facilement transmettre des informations à d'autres membres de notre équipe par le biais de graphique, de fonctions facilement applicable (somme, moyenne, etc.) .

14 Configuration Puppet

Les installations logicielles se font essentiellement à l'aide d'images PXE dans le cas des Nœuds ayant au moins un disque, et les images PXE des Nœuds diskless contiennent déjà tout ce dont ces Nœuds ont besoin. Le rôle de Puppet ici sera essentiellement :

- de faire des mises à jour des images PXE d'installation et diskless sur les Nœuds de service
- de faire des mises à jour logicielles à chaud sur chacun des Nœuds (il faudra pour cela créer un miroir des dépôts RHEL9 sur un ou plusieurs Nœuds de service, en utilisant celui de l'université)
- de s'assurer du bon fonctionnement des services
- d'ajouter de la configuration spécifique à certains Nœuds

Dans notre conception de l'architecture des serveurs Puppet, nous avons choisi de déployer l'un des serveurs sur un nœud maître. Cette décision découle du fait que les nœuds maîtres doivent avoir le contrôle sur le service en cas d'arrêt de celui-ci. De plus, nous attribuerons 2 nœuds de services à Puppet, en utilisant du failover, cela nous permettra d'installer de la résilience si le service puppet dysfonctionne sur l'un des nœuds de service.



Les services utilisés sur le cluster sont OpenSSH, NFS, Rsyslog, MariaDB, NTP, OpenLDAP, BIND (DNS), SLURM, Shinken, Puppet, DHCP, TFTP et Apache HTTP (ces 3 derniers sont surtout là pour permettre de boot en PXE). Chacun de ces services requiert un démon pour les serveurs, et certains (Rsyslog, NTP, OpenLDAP, SLURM, Shinken, Puppet) requièrent un démon pour les clients aussi.

Groupe/Rôle	Profiles	Nodeset	Commentaire
Nœuds AC	Slurmd, Noeuds en IB, Nœuds de calcul, client NTP, client Rsyslog, client DNS	sabatier[300-601]	
Nœuds AL	Slurmd, Noeuds en IB, Nœuds de calcul, client NTP, client Rsyslog, client DNS	sabatier[200-249]	
Nœuds AX	Slurmd, Noeuds en IB, Nœuds de calcul, client NTP, client Rsyslog, client DNS	sabatier[100-149]	
Nœuds Master	Shinken, Serveurs PXE, client LDAP, admin LDAP, Noeuds en IB, client NTP, serveur SSH, client Rsyslog, client DNS	sabatier[020-021]	
Nœuds de services	client NTP, serveur NTP, SlurmctldHost, MariaDB, Puppet, serveur LDAP, Noeuds en IB, Serveurs Rsyslog, DHCP, client DNS, serveur DNS, serveur DHCP, serveur TFTP, serveur HTTP	sabatier[040-049]	Seulement certains noeuds de services sont des serveurs NTP
Nœuds I/O	Noeuds en IB, client NTP, client DNS, serveur NFS	sabatier[060-069]	Les noeuds en IB sont reliés au stockage lustre ainsi qu'au cluster
Nœuds de login	Noeuds en IB, Noeuds "utilisateurs", serveur SSH, client NFS, client NTP, client DNS, client LDAP	sabatier[000-009]	

Pour les modules communautaires, nous pouvons utiliser le site forge puppet (ici), sur ce site, nous pouvons retrouver des modules en lien avec les services situés dans le tableau ci dessus.

Nous pouvons notamment retrouver les services suivants :

- Rsyslog

- ntp
- slurm
- ldap
- shinken
- ssh
- etc.

15 Environnement Utilisateur

Nous utiliserons Easybuild afin d'offrir les environnements logiciels voulus. Les environnements suivants nous sont demandés :

- un environnement de développement Intel complet
- un environnement de développement GNU de base
- un environnement de développement pour des codes accélérés par GPU/NVIDIA .
- Une version récente de MATLAB et MATLAB-Engine

Il n'est pas précisé, ce qui est entendu par "environnement complet". Nous supposons donc qu'il comprend tout ce qui peut être nécessaire, de proche ou de loin, au développement afin que l'utilisateur n'est jamais à devoir installer un paquet en lien avec l'environnement demandé.

15.1 Environnement de développement Intel complet

Pour commencer, l'installation de la toolchain GCCcore est essentielle, car elle sert de fondement pour les autres outils, comme indiqué dans le diagramme d'Easybuild (ici) . Par la suite, il sera nécessaire de faire attention à quel toolchain utilisé. En effet, les dépendances devront soit dépendre de la partie FOSS (Free and Open Source Software), soit de la partie GCC.

Ensuite, nous installerons la toolchain Intel (qui n'est pas FOSS), celle-ci comportera :

- leurs compilateurs C/C++/Fortan (faisant partie de leurs toolkit HPC (ici))
- leur libraire MPI
- binutils et gcc, qui seront la base de leurs compilateurs.
- leur librairie Math Kernel

La version la plus récente est actuellement la version 2023b de la toolchain "system".

En plus cette toolchain, nous en inclurons d'autres d'Intel que nous jugeons utiles pour le développement. Bien que ces outils ne soient pas spécifiquement liés au HPC, ils sont nécessaires pour créer un environnement de développement complet. Selon le site d'Intel, ces ajouts ne sont pas compris dans leurs toolchain de base.

- IntelPython, permettant d'améliorer les performances d'un code Python par le biais d'Anaconda
- IntelDALL, qui ajoute une optimisation lors de l'analyse de données, cela n'est pas directement lié à l'environnement de développement, mais de nombreux code peuvent en avoir besoin dans un milieu universitaire.
- IntelCuda afin d'ajouter NVIDIA CUDA aux compilateurs fourni de base.

Nous pouvons cependant remarquer que ces toolchains ne sont plus mis à jour depuis un certain temps (la mise à jour la plus récente est IntelCuda (2020b)). Si le centre est spécifiquement dédié au HPC, nous pourrons vérifier auprès de l'université si ces toolchains sont indispensables. Si tel est le cas, nous pourrions envisager de ne pas les installer par souci de sécurité.

De plus, dans la mise en place du cluster, il n'y a aucun équipement Intel, nous doutons de l'utilité d'avoir un tel environnement installé, nous pensons qu'il aurait été préférable d'avoir un environnement de développement AMD complet.

15.2 Développement GNU de base

Pour un environnement de développement GNU de base, il sera nécessaire d'installer la toolchain GNU, celle-ci comprend uniquement des compilateur GCC et binutils. Il est demandé un développement GNU de base, cela semble donc suffisant. Elle dépend de la toolchain "system" et la version la plus récente est la 5.1.0-2.25.

15.3 Développement pour codes accélérés par GPU/NVIDIA

Pour la partie code accélérés, nous installerons, la toolchain CUDA, contenant la CUDA-toolkit, permettant aux utilisateur d'avoir accès aux instructions nécessaire aux développements GPU. De plus, nous installerons également la toolchain CUDACompat, cela permettra aux utilisateurs ayant le besoin d'utiliser des versions de CUDA, plus ou moins récentes, de pouvoir le faire plus facilement, comme indique le site de NVIDIA (ici).

15.4 MATLAB et MATLAB-ENGINE

Enfin , afin d'avoir accès au version récente de MATLAB ainsi que de MATLAB-ENGINE nous installerons les toolchains, MATLAB, qui comporte l'environnement interactif demandé, ainsi que l'API Python nommé MATLAB-ENGINE. La version de MATLAB installé sera la plus récente, soit la 2023b de la toolchain "system".

Nous pouvons voir que MATLAB-ENGINE nécessite une toolchain différentes , GCCcore/11.2.0 ou foss/2017b. Nous n'avons pas installer de toolchain foss jusqu'a présent, cela nous permettra de choisir la version la plus récente. Cependant, la version la plus récente (GCCcore/11.2.0) n'a pas été mis à jour depuis 2021b-9.11.19, et il n'est pas précisé si celle-ci est prise en compte dans une autre toolchain. Elle risque donc de poser des problèmes de compatibilités si les versions ultérieures à GCCcore/11.2.0 ne sont pas rétrocompatibles.

Nous pouvons remarquer que la majorité des toolchains demandés par les utilisateurs sont estampillés sous le nom de la toolchain system dans la documentation Easybuild. Selon la documentation d'Easybuild (ici), cela indique que la dernière version sera installé.

15.5 Systèmes de fichiers utilisés

Le système de fichiers utilisés seront directement dans les nœuds de login, ceux-ci possèdent des disques durs, ils seront accessibles en lecture seule. Afin que les utilisateurs aient facilement accès à leurs environnements. Le système de fichier utilisé sera donc ext4. L'avantage de cette solution

est la localité des environnements, permettant donc un accès plus rapide, de plus, cela évitera de surcharger le réseau. L'inconvénient que nous voyons à cela sera la mise à jour qu'il faudra faire sur chaque machine.

Pour cela, nous proposons de mettre des "mirrors" sur certains nœuds de services contenant les paquets à mettre à jour. Il suffira alors d'utiliser un crontab demandant à puppet de faire des mises à jour régulièrement. De plus, cela permettra aux administrateurs d'approuver les versions et de facilement décliner une mise à jour.

16 Déploiement

La signification de l'acronyme LTS, veut dire, "Long Term Support". Cela indique qu'une version est supportée sur une longue période, avec des mises à jour régulière. Cette version est généralement la plus stable, ou les mises à jour sont principalement des mises à jour de stabilité et en lien avec la sécurité.

RHEL 9.0, 9.1, 9.2, 9.3, les cartes ConnectX-6 (celles fournies par AzkarHPC) ainsi que Lustre 2.12.9 et 2.15.2 sont supportés par le driver MOFED. Pour les cartes ConnectX6, il faut le firmware 20.39.3004 pour être compatibles avec le driver MOFED. Les versions de LTS de Lustre sont les 2.15.*, les versions LTS de RHEL 9 sont celles dont le dernier chiffre est pair, et le support de RHEL 9 pour les clients Lustre est introduit à partir de Lustre 2.15.2.

Nous pouvons remarquer que l'élément permettant de choisir ou non quel version de RHEL utilisé est MOFED qui n'est compatible qu'avec la version 2.12.9 ainsi que la version 2.15.2.

Ci-dessous, la matrice de compatibilité. Nous supposons que les versions de Lustres sont rétrocompatibles avec les anciennes versions de RHEL9. Si ce n'est pas le cas, nous remplaçons les checks par les points d'interrogation.

Version de Lustre	RHEL 9.0 (LTS)	RHEL 9.1	RHEL 9.2 (LTS)	RHEL 9.3	MOFED
2.12.9	✗	✗	✗	✗	✓
2.15.1 (LTS)	✗	✗	✗	✗	✗
2.15.2 (LTS)	✓	✗	✗	✗	✓
2.15.3 (LTS)	✓/?	✓	✓	✗	✗
2.15.4 (LTS)	✓/?	✓/?	✓/?	✓	✗

Table 1: Matrice de compatibilité Lustre - RHEL+MOFED

La version utilisée sera donc la version RHEL 9.0 (une version LTS) qui permet d'avoir MOFED (compatible avec le firmware 20.39.3004 pour les cartes IB), ainsi qu'une version LTS de Lustre (la version 2.15.2), et tous ces composants, matériels comme logiciels, sont compatibles entre eux.

Dans un premier temps, on placera sur le serveur PXE de l'Université des images d'installation pour tous les nœuds munis de disques (nœuds maîtres, service, routeurs, login, NFS), soit 31 nœuds. On peut déployer un script sur les switchs du backbone configurer temporairement des VLANs dessus pour chaque type de nœud et configurer le serveur PXE en trunk. Ainsi, lorsqu'on démarre les 31 nœuds à disques, ils récupéreront chacun, une image sur le serveur PXE, et celui-ci devinera quelle image il doit envoyer en se basant sur le VLAN d'où vient la requête. Ensuite, un second script sera déployé sur les switchs du backbone pour le remettre dans son état initial.

Puis, pour pouvoir démarrer les nœuds de partitions AC, AL et AX, il faut mettre en place un serveur PXE (avec un serveur DHCP dédié uniquement à rediriger vers PXE, la configuration IP étant statique sur le réseau IB) sur les nœuds de service, dont l'installation évoquée au paragraphe ci-dessus se chargera. Les nœuds de calcul (AC, AX, AL, soient 352 nœuds) récupéreront chacun, une image sur les nœuds de service pour boot en diskless. Afin d'éviter de surcharger les nœuds de service et d'accélérer le démarrage de l'intégralité des nœuds de calcul, il est également possible de faire en sorte que des nœuds de calcul servent temporairement de serveurs PXE eux-mêmes afin d'effectuer un déploiement en arbre.

17 Planification de l'installation du cluster

Dans l'appel d'offre, AzkharHPC indique qu'il y a 11 racks. Nous partons du principe que la société AzkharHPC ne fait pas le pont.

Les différentes étapes qu'il faudra utiliser pour l'installation d'un cluster sont :

- Installation matérielle
- Intégration avec le reste du cluster
- VABF (vérification d'aptitude au bon fonctionnement) qui s'effectue avec un environnement de test ainsi que des données de test
- VSR (vérification de service régulier) qui se déroule en production avec des données réelles
- Corrections

Nous ne connaissons pas le temps requis pour une VABF, VSR ainsi que pour la correction. Nous avons trouvé sur internet un appel d'offre de l'institut paris région pour leur acquisition d'une nouvelle baie de stockage. Dans ce document il est indiqué qu'une VABF durera, au plus, une semaine (ici). N'ayant pas trouvé de meilleures documents, celui-ci nous servira de référence pour estimer le temps nécessaire. Nous supposerons également qu'une VSR dure 1 semaine. Nous n'avons trouvé aucune information sur la durée d'une étape de correction sur Internet. Cependant cette partie, semble importante et peut être longue, nous supposerons qu'elle durera en moyenne 10 jours.

La société AzkharHPC ne peuvent livrer que 2 racks par semaine. De plus, ceux-ci livre à 8h00, nous ne savons pas combien de temps dure la livraison une fois sur place, nous partirons du principe que le matériel sera disponible l'après-midi de la livraison.

Les éléments indispensables pour commencer l'installation est la récéption des racks pour les noeuds des services ainsi que pour les switchs ethernet et infiniband. En effet, ceux-ci sont essentielles au bon fonctionnement d'un cluster.

Voici l'ordre de livraison qui sera imposé à la société AzkharHPC (1 sera livré en premier, 6 en dernier) :

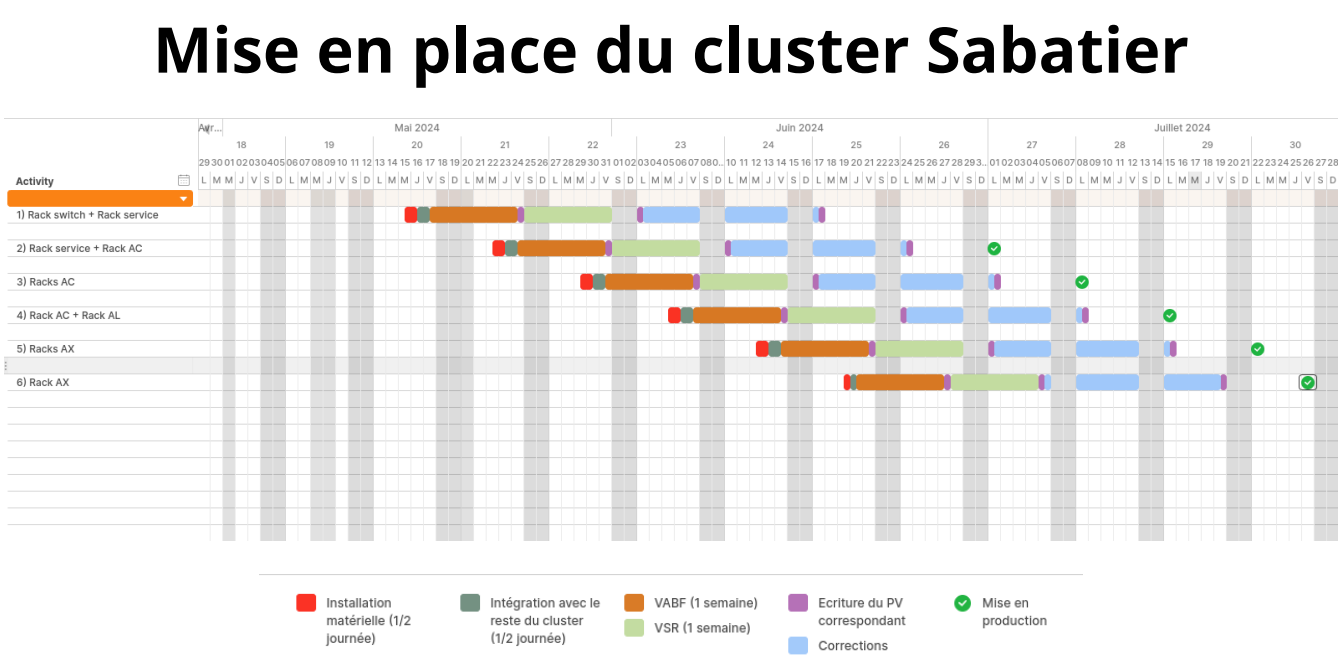
1. 1 rack pour les switchs ethernet et infiniband non intégrés dans les châssis AZ-B30 ; 1 rack pour les nœuds de service
2. 1 rack pour les nœuds de service ; 1 rack pour la partition AC
3. 2 racks pour la partition AC
4. 1 rack pour la partition AC ; 1 rack pour la partition AL
5. 2 racks pour la partition AX
6. 1 rack pour la partition AX

Lors de l'installation, nous avons décidé d'effectuer les 2 installations à la suite, et de ne pas profiter de la demi-journée qu'aurait un rack sur l'autre, cela permettra de les mettre en fonctionnement

au même moment. La VABF ainsi que la VSR, peuvent tourner le week-end, ce qui nous fera gagner du temps.

Dans ce scénario, nous supposons que nous sommes les personnes émettant les PV de recette provisoire, définitive ainsi que de levée de reserve. Nous nous accordons une demi-journée pour le faire. C'est une étape de validation, sans celle-ci, il est impossible de passer à l'étape suivante.

Il n'est également pas indiqué le temps de garantie, celui-ci pourrait nous pousser à modifier le Gantt, même si nous ne pensons pas qu'il soit possible d'aller plus rapidement que le Gantt proposé. Pour notre Gantt, nous supposons qu'il n'y a aucun problème et que toutes les étapes peuvent se suivre sans attente. Nous avons décidé de laisser 4 jours supplémentaires entre la fin de l'écriture du PV pour la correction et la mise en production. Cela permettra d'avoir un délai supplémentaire s'il y a des problèmes.



Finalement les utilisateurs auront accès à :

- Le 26 Juin 2024, 1 rack de la partition AC
- Le 03 Juillet 2024, 2 racks AC supplémentaires.
- Le 10 Juillet 2024, 1 rack AC et 1 rack AL.
- Le 17 Juillet 2024, 2 racks AX.
- Le 25 Juillet 2024, les utilisateurs auront accès au dernier rack du cluster, le rack AX.

Le cluster sera donc entièrement mis en place le 25 Juillet 2024. Nous indiquerons donc au utilisateurs que le cluster est mis en production progressivement en suivant ces dates de disponibilités :

- Partie AC disponible à partir du 26 Juin 2024.

- Partie AC entièrement disponible ; Partie AL entièrement disponible à partir du 10 Juillet 2024.
- Partie AX disponible à partir du 17 Juillet 2024.
- Partie AX entièrement disponible à partir du 25 Juillet 2024.