Telecom Paris

PRIM

Projet de recherche et d'innovation de mastère

Rapport

Optimisation d'une infrastructure de cloud computing par l'analyse prédictive des usages

Réalisé par :
Mohamed Elhedi Ben Yedder

Enseignant :
Olivier Fercoq

10 mars 2022

Table des matières

1 Introduction

Dans le cadre du programme du premier semestre à Telecom Paris, je dois effectuer un projet de recherche et d'innovation de master (PRIM). Le projet PRIM traite un sujet proposé par une laboratoire de l'école ou bien une entreprise partenaire.

1.1 Le sujet

Le sujet est celui d'optimisation d'une platforme cloud qui vise à améliorer l'algorithme d'équilibrage de la charge en se basant sur l'historique des noeuds et des machines virtuelles.
Le but est d'appliquer l'intelligence artificielle pour modéliser la charge et la consommation et proposer un nouveau algorithme amélioré. J'ai choisi un projet proposé par une startup pour découvrir l'environnement de travail en startups. En plus, j'étais toujours intéressé par le domaine du cloud et c'est une bonne opportunité pour découvrir de plus ce domaine et comprendre ces challenges.

2 Présentation du projet

2.1 Entreprise

FIGURE 1 – Logo de Weytop

Weytop est une entreprise créée en 2020 et ayant intégré l'incubateur en mars 2021. Elle vise à libérer les pratiques informatiques en les rendant plus nomades, plus écologiques, plus sécurisées et plus économiques. Elle développe et commercialise une solution innovante d'ordinateur personnel virtuel pour les entreprises. Ce "cloud computer" s'appuie notamment sur des technologies d'acquisition et de streaming très performantes. Weytop est aujourd'hui en phase d'accélération en déployant sa solution chez ses premiers clients, notamment dans le secteur de l'éducation formation.

2.2 Objectif du projet

L'objectif du projet PRIM est d'optimiser l'utilisation de l'infrastructure via l'intelligence artificielle en analysant les usages et les comportements des utilisateurs. La finalité du projet est d'améliorer l'overcommit, c'est à dire la capacité à mobiliser une machine pour plusieurs utilisateurs sans dégrader la performance perçue. L'objectif de ce projet est double :

— économique : il doit permettre de réduire les coûts d'hébergement
— écologique : il doit permettre d'optimiser l'empreinte environnementale du service

Sur ce dernier point, le projet sera couplé à une analyse de cycle de vie de la solution pour quantifier l'impact environnemental de la virtualisation en la comparant au coût carbone du renouvellement et de la gestion de parcs physiques traditionnels.

2.3 Problématique

La solution de weytop consiste à mettre à disposition de ses clients des machines virtuelles en streaming dans un navigateur. Ces machines virtuelles utilisent les capacités de virtualisation et de stockage d'un cluster de machines physiques. Lorsqu'un utilisateur souhaite accéder à sa machine virtuelle, cette dernière est affectée à une machine physique du cluster puis instanciée sur ce dernier. Le mécanisme d'affectation machine virtuelle <-> machine physique peut être optimisé avec des algorithmes plus évolués que la simple association naïve fonction du nombre de machines par machines physique. En effet, nous pourrions être dans le cas où 2xN machines virtuelles seraient instanciés sur 2 serveurs physiques A et B. Sur le serveur A, ce sont les N employés d'une société A qui se sont connectés quasi simultanément aux horaires d'ouverture des bureaux. Cet usage des machines virtuelles est très peu gourmand en ressources (CPU/RAM/Disk), les employés de la société A utilisant essentiellement le mail et la bureautique. En revanche, les employés sont connectés toute la journée pendant les heures de travail. A l'inverse, sur le serveur B, N machines virtuelles sont utilisées par des joueurs et des quelques employés d'un bureau d'étude de l'entreprise B. Ces machines virtuelles sont instanciés par intermittence tout au long de la journée et presque en permanence le soir de 19h00 à 01h30. Cet exemple illustre bien qu'une répartition plus fine en analysant le comportement de l'usage des machines virtuelles pourrait apporter un confort d'utilisation indéniable à iso ressources physiques.

3 Etat de l'art

3.1 Les bases de données RRD : Round-Robin-Database

La base de données "circulaires" (RRD) est un type de stockage très spécial conçu pour conserver les statistiques agrégées des séries chronologiques dans un fichier ou des tampons circulaires mappés en mémoire.

Étant donné que les données sont stockées dans une base de données circulaire basée sur un tampon, l'empreinte de stockage du système reste constante dans le temps.

La partie « round robin » de l'appellation provient de la structure de données de base utilisée pour stocker les points de données : les listes circulaires.

À court terme, chaque point de données est significatif et important : nous voulons une image précise de chaque événement qui s'est produit au cours des dernières 24 heures, ce qui peut inclure de petits pics transitoires d'utilisation du disque ou de la bande passante du réseau . Cependant, à long terme, seule une idée générale est nécessaire.

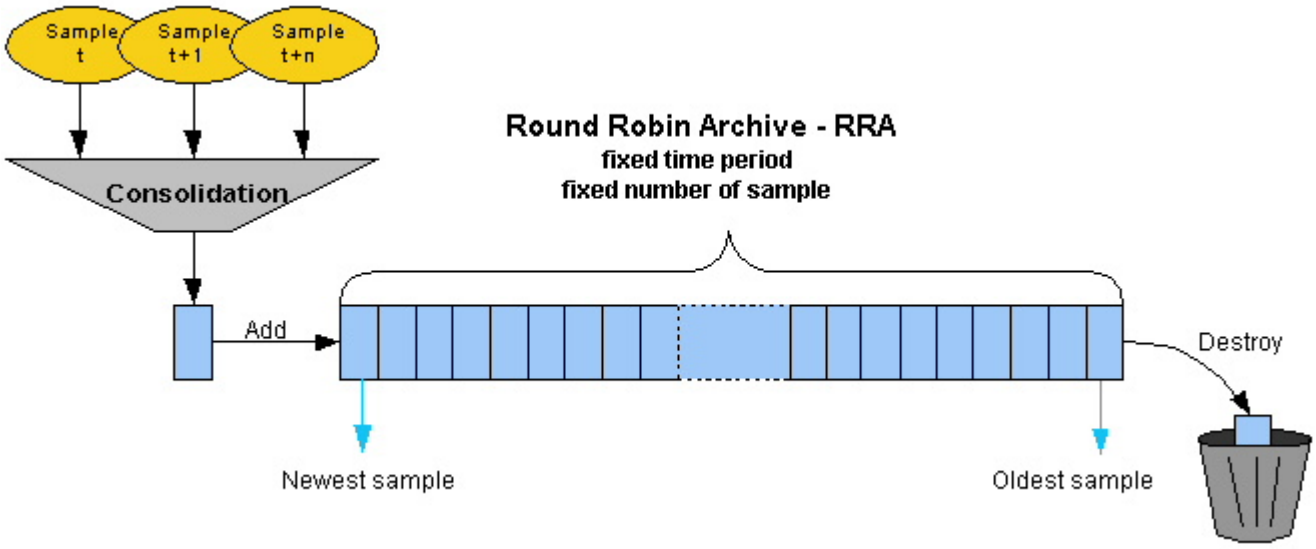


FIGURE 2 – Principe des bases de données RRD

3.2 Algorithmes d'apprentissage automatique

3.2.1 K-means

Le clustering est une méthode d'apprentissage non supervisé. Elle essaie de regrouper les données qui sont similaires sous une même classe(labelisation). Parmi les algorithmes de clustering K-means, c'est un algorithme non supervisé de clustering non hiérarchique. Il permet de regrouper les données en K clusters distincts. Les échantillons similaires sont regroupés sous un même clusteur. La clustorisation se fait en évaluant la similarité entre les données. Pour évaluer le degré de similarité, l'algorithme considère la distance séparant les données (distance euclidienne,distance de Hamming, ...) et il regroupe les données qui sont proches au sens de cette distance.

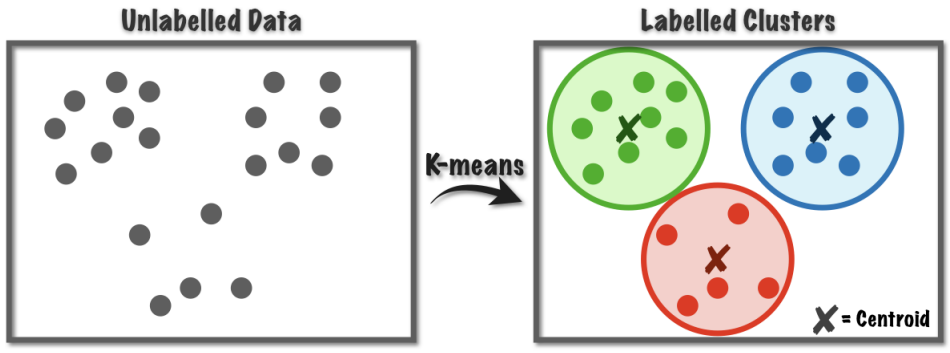


FIGURE 3 – Kmeans

3.2.2 Forêt d'arbres décisionnels (Forêt aléatoire)

C'est un algorithme d'apprentissage automatique. Cet algorithme combine les concepts de sous-espaces aléatoires et de bagging.
L'algorithme fait l'apprentissage de plusieurs arbres décisionnelles et les combine pour

réduire l'erreur(dilemme variance-biais).

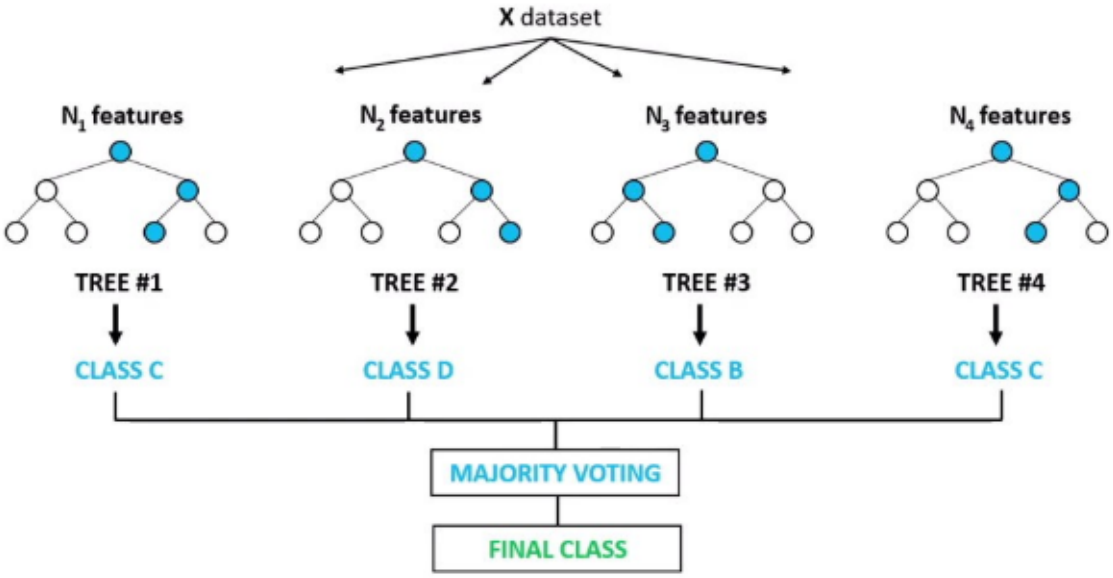


FIGURE 4 – Principe de Random Forest Classifier

4 Réalisation

4.1 Description des données

Tous les données sont stockées dans des fichiers ".rrd" (RRD : Round Robin Database) Nous disposons des données pour deux noeuds et sous chaque noeud nous avons :

- les données des chaque machines virtuelles qui ont été instansié sur ce noeud : Les données de chaque machine virtuelle sont enregistrées dans un fichier ".rrd" séparé
- les données des supports de stockage accessibles à travers le noeud : un fichier ".rrd" pour chaque support
- les données de consommation du noeud

Chaque fichier .rrd comporte des archives (RRA :Round Robin Archive) Chaque archive est définie par :

- un pas ou bien résolution (step) : période d'échantillonnage
- une fonction de consolidation (cf) : MAX | MIN |AVERAGE |LAST , à travers la fonction de consolidation , nous décidons si nous allons enregistrer le minimum , le maximum , la moyenne ou bien la dernière valeur (un simple échantillonnage)
- des attributs : les paramètres ou bien les caractéristiques que nous voulons enregistrer (par exemple : [consommation mémoire, consommation cpu , ...]).

Chaque archive (RRA) comporte exactement 75 échantillons exactement

RRD	Les atributs
Machine virtuelle (Consommation et Configuration)	maxcpu | cpu | maxmem | mem | maxdisk | disk | netin | netout | diskread | diskwrite | step
Support de stockage	total | used | step
Noeud (Utilisation et Resources disponibles)	loadavg | maxcpu |cpu iowait | memtotal | memused | swaptotal | swapused | roottotal | rootused | netin | netout | step

Les différents pas et fonctions de consolidation qui sont présents dans les différents archives (RRA) :

cf	MAX | AVERAGE
step	1m | 30m | 3h |12h| 3jrs |1 semaine

Nous ne pouvons pas agir sur ces archives pour modifier leurs paramètres . Ils sont définis initialement lors de l'initialisation de la base données. Une modification de ces paramètres implique la création de nouveaux archives càd ils représenteront les nouveaux paramètres pour l'enregistrement des nouvelles échantillons en futurs.

4.2 Structuration des données

Vu que les données sont stockées sous format ".rrd", il est nécessaire de les restructurer afin qu'elles seront exploitables. Nous allons donc définir des classes d'objets pour modéliser les différentes parties de l'infrastructure (Noeud, Machine virtuelle, Support de stockage, plat-forme)
Remarque : L'infrastructure actuelle comporte que deux noeuds , 209 machines virtuelles et 4 supports de stockages. Tous les classes d'objets et les algorithmes sont écrits de façons à considérer N_n noeuds et chaque noeud n_i peut accéder à $N_s(i)$ supports de stockage et sur lui sont instanciées $N_{vm}(i)$ machines virtuelles.

4.2.1 Classes d'objets : RRD et XMLRRD

Comme déja indiquée, les données sont stockées dans des fichiers ".rrd" D'où, nous avons crée deux classes d'objets pour les manipuler. Les classes d'objets RRD et XMLRRD seront la base de la construction des autres classes.

```
rrdfile=RRD("/content/gdrive/MyDrive/PRIM/Data/vmfarm3.weytop.com/db/pve2-node/airtop-vm-farm3")
rrdXmlfile=RRDXML(rrdfile,"file")
rrdXmlfile.toDataFrame(concate='all')
```

	Timestamp	loadavg_MAX	maxcpu_MAX	cpu_MAX	iowait_MAX	memtotal_MAX	memused_MAX	swaptotal_MAX	swapused_MAX	roo
1637154960s60	1637154960	5.830333	32.0	0.185966	0.000005	2.700643e+11	6.912730e+10	4.999606e+09	38797312.0	
1637155020s60	1637155020	5.832500	32.0	0.186355	0.000000	2.700643e+11	6.912544e+10	4.999606e+09	38797312.0	
1637155080s60	1637155080	5.675833	32.0	0.188997	0.000000	2.700643e+11	6.912621e+10	4.999606e+09	38797312.0	
1637155140s60	1637155140	5.854167	32.0	0.219738	0.000006	2.700643e+11	6.912537e+10	4.999606e+09	38797312.0	
1637155200s60	1637155200	6.609667	32.0	0.217228	0.000006	2.700643e+11	6.912603e+10	4.999606e+09	38797312.0	
...	...	...	...	...	...	...	...	...	...	...
1634169600s604800	1634169600	13.580000	32.0	0.413215	0.002177	2.700643e+11	1.080200e+11	4.999606e+09	38797312.0	
1634774400s604800	1634774400	17.328667	32.0	0.533089	0.002861	2.700643e+11	1.183041e+11	4.999606e+09	38797312.0	
1635379200s604800	1635379200	6.038333	32.0	0.189264	0.003097	2.700643e+11	3.586860e+10	4.999606e+09	38797312.0	
1635984000s604800	1635984000	6.176667	32.0	0.195271	0.002407	2.700643e+11	4.060693e+10	4.999606e+09	38797312.0	
1636588800s604800	1636588800	6.036000	32.0	0.199552	0.002746	2.700643e+11	5.491367e+10	4.999606e+09	38797312.0	

FIGURE 5 – Exemple d'utilisation des classes ***RRD*** et ***RRDXML***

4.2.2 Classe d'objet : VM

Statistiquement, Nous avons en totale 209 machines virtuelles distinctes qui ont été instanciées. Une machine virtuelle peut être instanciée sur n'importe quel noeud. Si une machine virtuelle a été instanciée sur plusieurs noeuds : Noeuds(VM_i). Les données d'une machine virtuelle VM_i lorsqu'elle est instanciée sur noeud n_k sont enregistrées dans un fichier ".rrd" $f_{i,k}$.Donc, les données d'une seule machine virtuelle VM_i peuvent être dispersées sur plusieurs fichier ".rrd" $f_{i,k}$, pour k $\in Noeuds(\mathrm{VM}_i)$.
À travers cette classe d'objet VM , nous arrivons à identifier les noeuds sur les quels une machine virtuelle VM_i a été instanciée : Noeuds(VM_i), localiser les fichiers ".rrd" correspondants et fusionner les données sous un format exploitable.

```
vm316=VM(316)
vm316.toDataFrame()
```

	Timestamp	maxcpu_MAX	cpu_MAX	maxmem_MAX	mem_MAX	maxdisk_MAX	disk_MAX	netin_MAX	netout_MAX	diskread_MAX	diskwrite_MAX	step	maxcpu_AVERAGE	cpu_AVERAGE	maxmem_AVERAGE
1632488100s60	1632488100	2.0	0.030051	4.294967e+09	3.651190e+09	3.435974e+10	0.0	1.287750e+02	12.883333	0.000000e+00	1.301589e+04	60	2.0	0.030051	4.294967e+09
1632488160s60	1632488160	2.0	0.019420	4.294967e+09	3.651181e+09	3.435974e+10	0.0	5.014167e+01	23.233333	0.000000e+00	2.005163e+04	60	2.0	0.019420	4.294967e+09
1632488220s60	1632488220	2.0	0.028800	4.294967e+09	3.651283e+09	3.435974e+10	0.0	1.753250e+02	24.133333	0.000000e+00	1.247386e+05	60	2.0	0.028800	4.294967e+09
1632488280s60	1632488280	2.0	0.018352	4.294967e+09	3.651283e+09	3.435974e+10	0.0	1.897750e+02	59.000000	0.000000e+00	6.417067e+03	60	2.0	0.018352	4.294967e+09
1632488340s60	1632488340	2.0	0.018818	4.294967e+09	3.651283e+09	3.435974e+10	0.0	1.355250e+02	29.083333	0.000000e+00	6.016000e+03	60	2.0	0.018818	4.294967e+09
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1634169600s604800	1634169600	2.0	1.031499	4.294967e+09	3.571430e+09	3.435974e+10	0.0	6.861783e+06	33694.350000	2.685328e+07	2.410243e+07	604800	2.0	0.031658	4.294967e+09
1634774400s604800	1634774400	2.0	1.040342	4.294967e+09	3.554659e+09	3.435974e+10	0.0	8.307499e+05	24035.411667	8.545159e+06	2.006191e+07	604800	2.0	0.028751	4.294967e+09

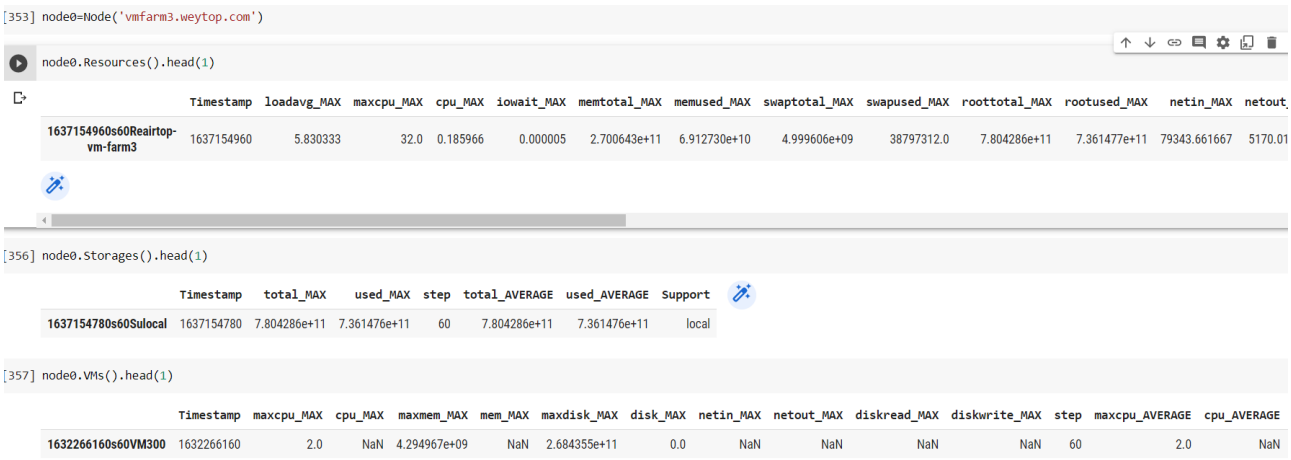
FIGURE 6 – Exemple d'utilisation de la classe ***VM***

4.2.3 Classe d'objet : Noeud

Nous avons seulement deux noeuds. Les données correspondantes à chaque noeuds sont dispersées sur plusieurs fichiers ".rrd" :

- Les fichiers ".rrd" des machines virtuelles qui ont été instanciées sur le noeud.
- Les fichiers ".rrd" des supports de stockage accessibles par le noeud
- Le fichier ".rrd" de la charge (utilisation RAM/CPU ..) du noeud

À travers cette classe d'objet Noeud , nous arrivons à représenter ces informations d'une façon cohérente et sous un format de données exploitable.



```
[353] node0=Node('vmfarm3.weytop.com')

node0.Resources().head(1)
```

	Timestamp	loadavg_MAX	maxcpu_MAX	cpu_MAX	iowait_MAX	memtotal_MAX	memused_MAX	swaptotal_MAX	swapused_MAX	roottotal_MAX	rootused_MAX	netin_MAX	netout
1637154960s60Reairtop-vm-farm3	1637154960	5.830333	32.0	0.185966	0.000005	2.700643e+11	6.912730e+10	4.999606e+09	38797312.0	7.804286e+11	7.361477e+11	79343.661667	5170.01

```
[356] node0.Storages().head(1)
```

	Timestamp	total_MAX	used_MAX	step	total_AVERAGE	used_AVERAGE	Support
1637154780s60Sulocal	1637154780	7.804286e+11	7.361476e+11	60	7.804286e+11	7.361476e+11	local

```
[357] node0.VMs().head(1)
```

	Timestamp	maxcpu_MAX	cpu_MAX	maxmem_MAX	mem_MAX	maxdisk_MAX	disk_MAX	netin_MAX	netout_MAX	diskread_MAX	diskwrite_MAX	step	maxcpu_AVERAGE	cpu_AVERAGE
1632266160s60VM300	1632266160	2.0	NaN	4.294967e+09	NaN	2.684355e+11	0.0	NaN	NaN	NaN	NaN	60	2.0	NaN

FIGURE 7 – Exemple d'utilisation de la classe ***Noeud***

4.2.4 Classe d'objet : Platform

La classe d'objet Platform modélise la totalité du système :

- Les noeuds
- les machines virtuelles
- les supports de stockage

```
[358] import rrdtool as rrd
      weytop=Platform("weytop",description="Platform")

[359] start,end,step=weytop.start(),weytop.end(),60

[360] start,end,step

      (1616929620, 1637159100, 60)

[361] fromtimestamp(start),fromtimestamp(end),step

      ('2021-03-28 11:07:00', '2021-11-17 14:25:00', 60)

      print(weytop)

      ----------------------------------------------
      Node0: vpn1.airtop.io
      VMs :['100', '101', '102', '104', '107', '111', '131', '132', '133', '134', '135', '136', '137', '138', '139', '140', '141', '142', '143', '144', '145', '146', '147', '148', '149', '
      Storage Supports: ['local', 'vm_disk_storage', 'local']
      Resources: ['vpn1', 'airtop-vm-farm1']
      Journals: ['rrd.journal.1637154355.422241', 'rrd.journal.1637157955.422251']

      ----------------------------------------------
      Node1: vmfarm3.weytop.com
      VMs :['100', '101', '102', '103', '104', '105', '106', '107', '108', '109', '110', '111', '112', '113', '200', '201', '300', '305', '306', '309', '310', '311', '312', '313', '314', '
      Storage Supports: ['local', 'vmstorage', 'vmfarm3storage1']
      Resources: ['airtop-vm-farm3']
      Journals: ['rrd.journal.1637153273.680935', 'rrd.journal.1637156873.680960']
```

FIGURE 8 – Exemple d'utilisation de la classe Platform

4.2.5 Problèmes rencontrées avec les données

Problème	Solution et commentaire
Les données sont distribuées sur plusieurs fichiers	Utilisation des classes d'objet pour les structurer
Les données sont enregistrées avec plusieurs pas (fréquence d'échantillonnage)	Séparation des données selon le pas
Impossible de superposer les différentes données des différents objets (Machines virtuelles/ Nœuds / Supports du stockages) car l'instant de début d'enregistrement des données pour un objet est aléatoire.	décaler les TimeStamp avec la formule suivante : t = arrondi(t/pas)*pas valable pour les pas de valeur faible seulement.
Nous ne pouvons pas vérifier est ce que la consommation d'un noeud à un instant est égale à la consommation des VMs qui sont instanciés sur lui (à cause du décalage des instants d'échantillonnage) La consommation d'un noeud n'est pas égale à la somme de la consommation des machines virtuelles instanciées sur ce noeud à une constante prés (si nous négligeons le décalage des instants d'échantillonnage)	Nous avons essayé de vérifier la somme des consommation des VMs et la consommation d'un noeud après le décalage des instants pour un pas = 60s , Solution : Modélisation de la consommation

4.3 Critère d'évaluation de la distribution de la charge sur les noeuds

La charge supportée par un noeud peut être représentée par un vecteur X = (CPU , Mémoire , Réseau,....) le critère porte sur la minimisation de la variance. à un instant t , nous calculons la variance de la distribution de la charge sur les tous les noeuds

- si l'écart type de la variable choisie est très grand (par exemple :la mémoire) nous avons une distribution non uniforme de point de vue cette variable
- si l'écart type de la variable choisie est très petit (tend vers 0) nous avons une distribution quasi-uniforme

	MEM_mean	CPU_mean	netin_mean	netout_mean	MEM_std	CPU_std	netin_std	netout_std	MEM_node	CPU_Nodes	netin_node	netout_node
1616929620	2.610687	0.000000	1.747025e+05	5.073333	3.692069	0.000000	2.470666e+05	2470.666449	1=>0	0=>1	1=>0	1=>0
1616929680	2.610791	0.000000	2.210367e+05	0.526667	3.692216	0.000000	3.125931e+05	3125.930518	1=>0	0=>1	1=>0	1=>0
1616929740	2.610002	0.000000	1.885030e+05	4.283333	3.691100	0.000000	2.665835e+05	2665.834991	1=>0	0=>1	1=>0	1=>0
1616929800	2.610292	0.000000	1.562687e+05	0.000000	3.691511	0.000000	2.209973e+05	2209.972678	1=>0	0=>1	1=>0	0=>1
1616929860	2.610554	0.000000	1.791633e+05	4.283333	3.691880	0.000000	2.533752e+05	2533.752159	1=>0	0=>1	1=>0	1=>0
...	...	...	...	...	...	...	...	...	...	...	...	...
1637158860	51.204827	5.431000	1.145022e+07	5525.711667	36.216493	1.269021	1.384594e+07	138459.424998	1=>0	0=>1	0=>1	0=>1
1637158920	12.797679	3.137500	1.054849e+07	2889.175000	18.098651	4.437095	1.491782e+07	149178.211561	0=>1	0=>1	0=>1	0=>1
1637158980	12.797888	3.129167	1.044338e+07	2624.366667	18.098947	4.425310	1.476917e+07	147691.731685	0=>1	0=>1	0=>1	0=>1
1637159040	12.797976	3.115833	1.057341e+07	3028.883333	18.099071	4.406454	1.495306e+07	149530.633581	0=>1	0=>1	0=>1	0=>1
1637159100	12.797815	3.148333	6.468289e+06	4662.316667	18.098843	4.452416	9.147542e+06	91475.422646	0=>1	0=>1	0=>1	0=>1

FIGURE 9 – Evaluation de la distribution actuelle des données par rapport à chaque variable : Les colonnes -node correspondent au choix du noued sur le quel il faut instancier la machine virtuelle à l'instant t+1

La limite de ce critère que nous devons toujours choisir une variable (CPU , Mémoire, ...) et optimiser la distribution selon cette variable en négligeant les autres

Pour avoir un critère globale qui prend en considération toutes les variables, à chaque écart type σ_v relative à une variable $v, v \in$ { CPU , MEM , NetIN , NetOut ....} nous associons au début :

$$\sigma_v{}' = \sigma_v/(max(v) - min(v)) \tag{1}$$

Ce terme $\sigma_v{}'$ est sans unité. Il varie toujours entre 0 et 1 et nous pouvons l'interpréter comme un écart-type normalisé.
Ensuite, nous considérons une moyenne pondérée des $\sigma_v{}'$:

$$\bar{\sigma'}(\alpha_1, \alpha_2, ...\alpha_M) = \sum_{i=1}^{M}(\alpha_i * \sigma_{v_i}) \tag{2}$$

Les coefficients α_i sont choisis selon l'importance des variables v_i.
Le but de l'introduction de ces coefficients est d'avoir une plat-forme paramétrée de façon que le responsable dans l'entreprise de la tâche d'équilibrage de la charge aie la possibilité de la configurer selon les préférences de l'entreprise.

4.4 Modélisation

Tous les modèles dans cette parties sont implémentés avec la bibliothèque libre de python scikit-learn.

4.4.1 Modélisation de la charge d'un noeud

Nous avons indiqué, après une vérification au niveau des données, que la charge d'un noeud n'est pas égale à la somme de la consommation des différentes machines virtuelles instanciées sur ce noeud (à une constante prés).
Nous voulons modéliser la charge d'un noeud pour prévoir sa charge lorsque une nouvelle machine est instanciée sur lui.
Nous allons considérer un noeud N sur lequel sont instanciées p machines virtuelles.
$I_p = \{vm_1, ..vm_p\}$ est l'esnsemble des machines virtuelles instanciées sur ce noeud.
Nous allons instancier une nouvelle machine virtuelle vm_{p+1} sur ce noeud N et nous estimons que nous connaissons la consommation de cette nouvelle machine virtuelle vm_{p+1} , $I_{p+1} = I_p U \{vm_{p+1}\}$

$$C(N, I_{p+1}, t_{p+1}) = C(N, I_p, t_{p+1}) + aC(vm_{p+1}, t_{p+1}) + b \tag{3}$$

où a, b : les paramètres du modèle, $C(N, I, t)$: la charge du noeud N lorsque toutes les machines de I sont instanciées sur lui à l'instant t ,$C(vm, t)$: la consommation de la machine virtuelle vm à l'instant t ,t_{p+1} : l'instant de l'instanciation de vm_{p+1}.
Le problème que nous ne pouvons pas faire l'apprentissage avec cette formule pour retrouver les coefficients a et b pour quelques raisons. En effet, pour avoir des résultats précis , il faut se limiter aux données enregistrées avec un pas très faible (pas = 60 secondes) et

sur ces données , les instants où il y a une instanciation d'une nouvelle machine virtuelle sont limités.
La solution est de considérer un autre modèle qui est indépendant des instants d'instanciation et identifier les coefficients a et b à partir de ce modèle : Nous considérons donc ce nouveau modèle :

$$C(N, I_{p+1}, t) = \alpha \sum_{vm \in I_{p+1}} C(vm, t) + \beta(p+1) + \gamma \tag{4}$$

où α, β, γ : les paramètres du modèle, $p+1$: le nombre des machines virtuelles
Comparé au modèle précédent, ce modèle essaie de de retrouver la relation entre la charge du noeud et la somme de la consommation des machines virtuelles instanciées sur lui à un instant t. L'apprentissage peut être fait sur la totalité des données (tous les instants non pas seulement les instants d'instanciation)
Une fois que les coefficients α, β, γ sont trouvés, nous pouvons identifeir les coefficient a,b en développant l'expression :
$C(N, I_{p+1}, t) = \alpha \sum_{vm \in I_{p+1}} C(vm, t) + \beta(p+1) + \gamma$
$C(N, I_{p+1}, t) = \alpha(\sum_{vm \in I_p} C(vm, t) + C(vm_{p+1}, t)) + \beta(p+1) + \gamma$
$C(N, I_{p+1}, t) = (\alpha \sum_{vm \in I_p} C(vm, t) + \beta(p) + \gamma) + \alpha C(vm_{p+1}, t) + \beta$
$C(N, I_{p+1}, t) = C(N, I_p, t) + \alpha C(vm_{p+1}, t) + \beta,$
Regardons cette expression à l'instant t_{p+1} de l'instanciation de la machine virtuelle vm_{p+1} :

$$C(N, I_{p+1}, t_{p+1}) = C(N, I_p, t_{p+1}) + \alpha C(vm_{p+1}, t_{p+1}) + \beta \tag{5}$$

D'aprés (3) et (5), nous retrouvons $a = \alpha, b = \beta$

4.4.2 Modélisation de la consommation d'une machine virtuelle

Le but est de prévoir la consommation habituelle de la machine virtuelle au moment de son instanciation.
Nous allons considérer qu'un utilisateur utilise sa machine virtuelle pour k activités différentes en terme de consommation {0,1,..,k}.
Au début, nous allons identifier les k classes d'utilisation (clustering) et ensuite ,créer un modèle capable de prévoire la classe au moment de l'instanciation.
remarque : Le clustering et la modélisation seront appliqués à chaque machine virtuelle (historique d'un utilisateur) séparément. Autrement dit, chaque machine virtuelle a sa propre k clusteurs et son propre modèle de prédiction

— **Clustering**

Nous avons choisi de travailler sur les données enregistrés avec un pas = 3 heures. le choix du pas n'est pas aléatoire. Avec un pas = 3 heures, nous pouvons couvrir toutes les périodes de la journées (24/3+1=9 périodes dans la journées) Nous avons utilisé l'algorithmes Kmeans pour identifier les k clusteurs. L'hyperparamètre k est fixé.
nous avons fixer k = 3 en se basant sur notre propre utilisation habituelle de l'ordinateur.

— **Modélisation**

Dans cette étape, nous avons crée un modèle capable de prévoir la classe (le clusteur) à partir de l'instant de d'instanciation de la machine virtuelle ("hour","day","weekday","isweekend"). L'algorithme utilisé est l'algorithme de forêt d'arbres décisionnels (forêt aléatoire). Les hyperparamètres sont initialisés par défaut.

4.4.3 Algorithme de décision

L'algorithme actuel propose d'instancier chaque fois la nouvelle machine virtuelle sur un nouveau noeud en alternant (permutation circulaire :modulo). Nous proposons un autre algorithme qui tend à avoir une distribution uniforme de la charge selon nos préférences (coefficients de pondération).
Soit une machine virtuelle vm_i à instancier à l'instant t.

1. A travers la modèle de consommation d'une machine virtuelle (4.4.2), nous faisons la prédiction de la consommation habituelle de la machine virtuelle vm_i : $C(vm_i)$
2. Pour chaque noeud N,
   - nous supposons que vm_i sera instanciée sur lui et nous calculons la prédiction de sa charge ($C'(N) = C(N) + a * C(vm_i) + b$ équation (5)) et pour les autres noeuds nous considérons la prédiction de la consommation est égale à sa charge actuelle ($C'(N) = C(N)$)
   - Par rapport à chaque variable v (CPU, Mémoire ,...), nous calculons l'écart-type 'normalisé de la charge par rapport à v sur tous les noeuds $\sqrt(var(C'[v]))/(max(v) - min(v))$ équation (1)
   - nous calculons la moyenne pondérée des écarts-type (normalisés) sur toutes les variables v : $\sigma'[N] = \sum_v \alpha_v * \sigma_v$
3. Nous instancions vm_i sur le noeud qui minimise σ'

Algorithm 1 Algorithme de décision

for $N \in N_1, ..., N_m$ **do** ▷ Initialisation
  $\sigma'[N] = 0$
end for
$C(vm_i) \leftarrow f_{vm_i}(t)$ ▷ $f_{vm_i}(t)$ c'est le modèle de prédiction 4.4.2
for $N \in N_1, ..., N_m$ **do**
  $C'(N) = C(N) + a * C(vm_i) + b$ ▷ C(N) : la charge actuelle du noeud N, voir l'équation (5)
  $C'(N') = C(N') \; \forall \; N' \neq N$ ▷ C et C' sont deux vecteurs chaque composante représente la consommation par rapport une variable (CPU,Mémoire,...)
  for $v \in \{CPU, NetIN, NetOUt, ..\}$ **do**
    $\sigma_v = \sqrt(var(C'[v]))/(max(v) - min(v))$
    $\sigma'[N] += \alpha_v * \sigma_v$ ▷ α_v sont les coefficient de pondérations
  end for
end for
$decision \leftarrow arg_N min(\sigma')$ ▷ on choisit le noeud qui minimise σ'

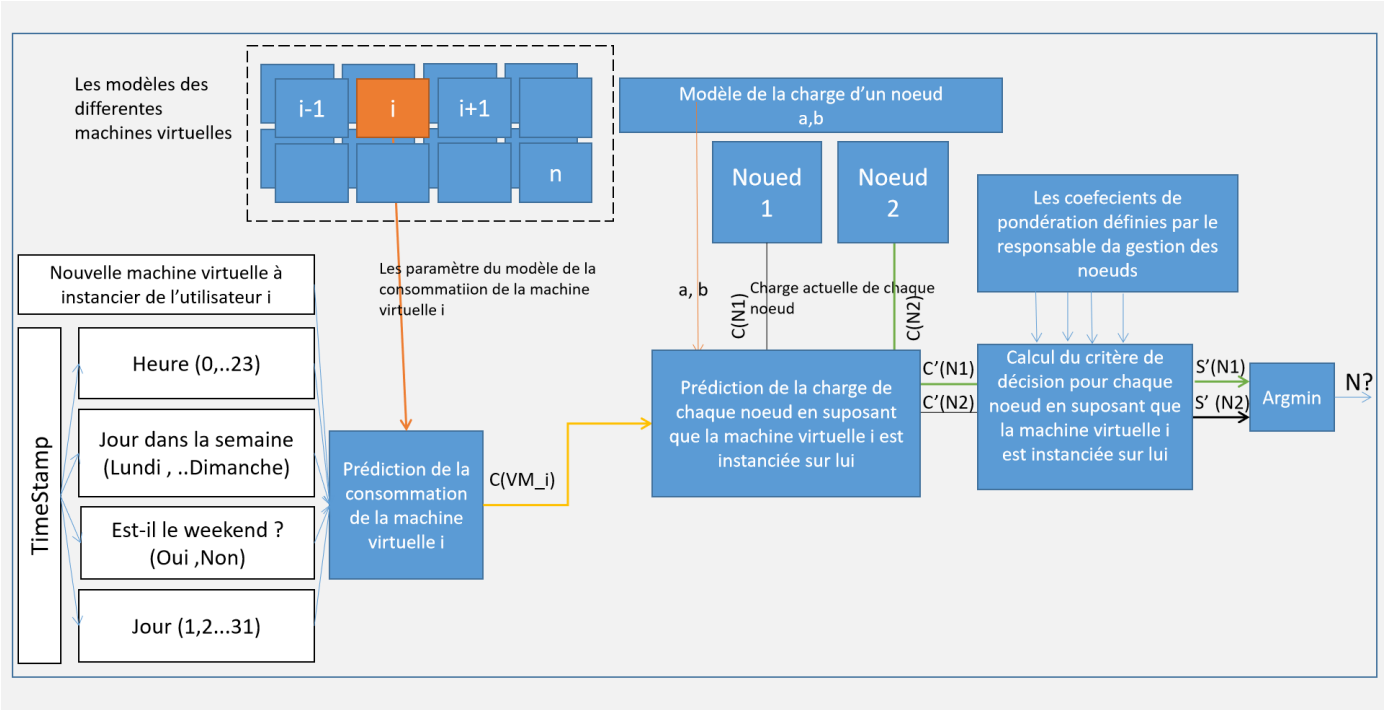


FIGURE 10 – Graphe expliquant l'algorithme

5 Résultats

5.1 Évaluation du modèle de prédiction de la charge d'un noeud

Pour chaque variable $v \in \{CPU, RAM, NetIN, Netout\}$, nous avons crée un modèle linéaire :

$$C(N, I_{p+1}, t_{p+1}) = C(N, I_p, t_{p+1}) + aC(vm_{p+1}, t_{p+1}) + b \tag{6}$$

Nous avons retrouver les résultats suivants :

Variable	a	b
RAM	0.99585645	-545.52709379
CPU	1.85953154	-0.34058604
netIN	0.845185	-730.66786327
netOUT	0.99585645	-545.52709379

Pour évaluer la performance du modèle, nous avons calculer une métrique d'évalution : $R = 1 - eqm(y_{pred})/var(y_{true})$
où $var(y_{pred})$ est l'erreur quadratique moyenne de prédiction et $var(y_{true})$ est la variance des valeurs réelles à prédire. cette metrique peut avoir des valeurs négative, le cas idéale est d'avoir une valeur trés proche de 1.

Variable	R
RAM	-6.5697299777050615
CPU	0.935705329122057
netIN	0.9505782370514909
netOUT	0.7888001188592341

5.2 Évaluation du modèle de prédiction de la consommation d'une machine virtuelle

Pour chaque machine virtuelle, nous avons crée un modèle qui prédit sa consommation à l'instant de son instanciation. Il y a quelques machines virtuelles qui nous ne pouvons pas leurs associer un modèle car les données décrivant la consommation de ces machines virtuelles sont trés limitées en quantité.([208, 228, 231, 242, 159, 229, 316, 254, 235]). L'évaluation montre le taux des prédictions correctes dépassent toujours 0.5 pour tous les machines virtuelles

6 Conclusion

Durant le projet nous avons accompli plusieurs tâches :

- Structuration des données et modélisation de chaque éléments (Noeud, VM, Support de stockage, plat-forme) du problème par des objets.
- Proposition d'un critère d'évaluation paramétré qui peut être configurer selon les préférences de l'entreprise
- Modélisation de la charge d'un noeud
- Modélisation de la consommation d'une machine virtuelle
- Proposition d'un algorithme de décision alternative basé sur les modèles créés et le critère d'évaluation

L'évaluation des modèles a montré de bons résultats. Dans le future, nous pouvons travailler sur l'apprentissage des hyperparamètres pour améliorer les résultats.(surtout le nombre de clusteurs)
L'évaluation de l'algorithme de décision nécessite l'implémention de l'algorithme sur la platforme réelle et l'évaluation de la réponse selon le critère d'évaluation

PRIM

10 mars 2022

Réferences
(1) La bibliothèque de scikit-learn [scikit-learn.org]
(2) Le code (implémentation et données) [drive]