Meilleure élasticité « nuagique » par commande sans modèle

Improving resource elasticity in cloud computing thanks to model-free control

Maria Bekcheva[1], Michel Fliess[2,4], Cédric Join[3,4], Alireza Moradi[5], Hugues Mounier[1]

[1]Laboratoire des Signaux et Systèmes (L2S), Université Paris-Sud-CNRS-CentraleSupélec, Université Paris-Saclay, 91192 Gif-sur-Yvette, France, {maria.bekcheva, hugues.mounier}@l2s.centralesupelec.fr
[2]LIX (CNRS, UMR 7161), École polytechnique, 91128 Palaiseau, France, Michel.Fliess@polytechnique.edu
[3]CRAN (CNRS, UMR 7039), Université de Lorraine, BP 239, 54506 Vandœuvre-lès-Nancy, France, cedric.join@univ-lorraine.fr
[4]AL.I.E.N. (ALgèbre pour Identification & Estimation Numériques), 7 rue Maurice Barrès, 54330 Vézelise, France, {michel.fliess, cedric.join}@alien-sas.com
[5] Inagral, 128 rue de la Boétie, 75008 Paris, France, alireza@inagral.com

RÉSUMÉ. L'adaptation dynamique des ressources de calcul à des variations de trafic, dans la gestion « nuagique », est un domaine actif d'investigation. Les automaticiens ont déjà proposé maints remèdes. On emploie, ici, la commande sans modèle et les correcteurs « intelligents » associés, faciles à implanter et aux nombreux succès industriels, pour traiter l'« élasticité horizontale ». Le comportement, comparé aux algorithmes commerciaux d'auto-ajustement, est meilleur, même avec des fluctuations aigües de charge. Des expériences sur le service Web d'Amazon (AWS) le confirment.

ABSTRACT. In cloud computing management, the dynamic adaptation of computing resource allocations under time-varying workload is an active domain of investigation. Several control strategies were already proposed. Here the model-free control setting and the corresponding "intelligent" controllers, which are most successful in many concrete engineering situations, are employed for the "horizontal elasticity." When compared to the commercial "Auto-Scaling" algorithms, our easily implementable approach, behaves better even with sharp workload fluctuations. This is confirmed by experiments on Amazon Web Services (AWS).

MOTS-CLÉS. Nuagique, allocation des ressources de calcul, ajustement, élasticité, commande sans modèle, AWS.
KEYWORDS. Cloud computing, computing resources allocation, scaling, elasticity, model-free control, AWS.

This is not the end. It is not even the beginning of the end.
But it is, perhaps, the end of the beginning.
Churchill (10 novembre 1942)

1. Introduction

1.1. *Prolégomènes*

Nul besoin d'acquérir et de maîtriser matériels et logiciels chers. L'essentiel se fait ailleurs, via Internet. C'est le but du « nuagique », raccourci d'« infonuagique », inventé au Québec pour traduire *cloud computing* [1]. Voici quelques échantillons [Armbrust *et coll.* (2010), Buyya *et coll.* (2013)], [Fehling *et coll.* (2014), Marinescu (2017), Wu & Buyya (2015)] de la vaste littérature consacrée à cette technologie informatique en croissance rapide. Un marché considérable se développe. Amazon, Google, Microsoft, IBM, Oracle aux États-Unis, Alibaba, China Telecom, Tencent en Chine, NTT Communica-

1. La presse française écrit souvent « informatique dématérialisée ». C'est long et, surtout, trompeur. Un matériel lourd est toujours là, mais loin de l'utilisateur.

tions au Japon, SAP en Allemagne, Orange, OVH en France sont parmi les entreprises les plus connues. La faculté d'ajuster au mieux et en temps réel les ressources de calcul réclamées par l'usager, qui est un avantage clé du nuagique, s'appelle « élasticité » (voir, par exemple, [Herbst *et coll.* (2013)]). Ce sujet très actif de recherches est, ici, le nôtre.

Avant de passer au vif, quelques remarques s'imposent :

— Écrire en français est une gageure. La domination absolue de l'anglais lui a donné un statut subalterne, notamment dans le champ scientifique. S'ajoute, dans une discipline neuve, la difficulté, considérable, de trouver des équivalents du vocabulaire américain [2]. Que l'impact de cette contribution n'en soit point trop amoindri !

— Kalman (voir [Kalman *et coll.* (1969)] et, aussi, [Eilenberg (1974), Sontag (1998)]) avait tenté d'établir un lien entre automatique, c'est-à-dire, alors, systèmes linéaires de dimension finie, et théorie des automates finis, adonc dominante en informatique théorique. Le lien, vieux de près de soixante ans [Schützenberger (1961)] entre automates finis, monoïdes libres et séries rationnelles non commutatives (voir, aussi, [Berstel & Reutenauer (2008), Sakarovitch (2003)]) rattache, en fait, ces automates aux systèmes « bilinéaires », ou « réguliers », de l'automatique grâce aux séries génératrices non commutatives rationnelles [Fliess (1978)]. Cette problématique, aussi valable soit-elle, a vieilli. L'interaction entre informatique et automatique a migré vers des sujets plus concrets (voir, par exemple, [Hellerstein *et coll.* (2004), Janert (2014), Leva *et coll.* (2013), Marinescu (2017)]). L'investigation des rapports entre automatique et informatique demeure néanmoins mineure à l'université et dans l'industrie.

— Voici un rajout quelque peu sarcastique. L'intelligence artificielle, qu'on ne sait définir vraiment, est vue aujourd'hui comme l'acmé de l'informatique, non seulement dans les médias, mais aussi par moult acteurs politiques, économiques, administratifs et universitaires. D'aucuns (voir, par exemple, [Chong (2017)] et sa bibliographie), dans des cercles d'ingénieurs, l'appréhendent comme un avatar de l'automatique.

1.2. *Rudiments d'élasticité*

Afin de répondre à des variations, parfois brutales, de charges, l'approche traditionnelle provisionne à l'avance, au risque d'un fort gaspillage, des ressources considérables. En nuagique, la puissance de calculs est liée aux « machines virtuelles », ou *virtual machines* (*VM*), c'est-à-dire à des émulations d'ordinateurs dues à des logiciels. Elles sont hébergées sur des serveurs physiques mutualisés. Un regroupement de machines virtuelles est appelé « grappe », ou *cluster*. L'« élasticité horizontale » modifie la cardinalité de la grappe sans en modifier les éléments [3].

Comme en témoignent les synthèses dues à [Al-Dhuraibi *et coll.* (2018), Galante *et coll.* (2012)], [Lorido-Botran *et coll.* (2014), Patikirikorala *et coll.* (2012), Ullah *et coll.* (2018)], de multiples techniques de commande ont été proposées pour assurer un meilleur « ajustement », ou *scaling*. Les PID, primordiaux dans l'industrie classique (voir, par exemple, [Åström & Hägglund (2006)], [Åström & Murray (2008), O'Dwyer (2009)]) y dominent aussi. L'explication reste identique : impossi-

2. Les publications en français sont rares (voir, par exemple, [Rivard (2012), Vicat-Blanc Primet *et coll.* (2010)]). La situation des autres « grandes » langues scientifiques occidentales, comme l'allemand, est similaire.

3. Quant à l'« élasticité verticale », elle altère la structure des machines virtuelles et/ou physiques

bilité d'une description mathématique exploitable pour la plupart des applications concrètes, quel qu'en soit le domaine.

1.3. *Notre approche*

On utilise la « commande sans modèle », ou *model-free control* (*MFC*), [Fliess & Join (2013)]. Elle
- conserve les avantages des PID sans en avoir les graves inconvénients [Fliess & Join (2013)], [Fliess & Join (2018)],
- a connu nombre de succès concrets (voir une liste assez complète, du moins jusqu'au début 2018, dans les bibliographies de [Fliess & Join (2013), Bara *et coll.* (2018)]),
- est facile à implanter [Fliess & Join (2013), Join *et coll.* (2013)].

Remarque 1. *On trouve en [Fliess & Join (2013)] des références sur l'emploi du vocable « sans-modèle » en automatique, mais avec des sens tout différents. Il en va de même en nuagique (voir, par exemple, [Bu et coll. (2013), Rao et coll. (2010), Wang et coll. (2012)]).*

L'algorithme est testé à l'aide d'*Amazon Web Services* ; l'acronyme *AWS* est connu de la plupart [4]. Les comparaisons avec deux démarches :
- sans « auto-ajustement », ou *auto-scaling*,
- et, surtout, avec l'« ajustement pour suivi de cible », ou *target tracking scaling*, d'Amazon *Elastic Compute Cloud* (*EC2*),

tournent largement en faveur du sans-modèle (voir tableau **4.1**).

1.4. *Plan*

On trouve des rappels sur cette commande au paragraphe 2.. Les paragraphes 3. et 4. détaillent respectivement mise en œuvre, expérimentations et comparaisons. La conclusion au paragraphe 5. suggère non seulement de futures pistes mais aussi des réflexions quant à la recherche nuagique.

4. Renvoyons, pour toute précision utile ici, à [Wittig & Wittig (2016)]) et, surtout, au lien, fourni par Amazon, `https://docs.aws.amazon.com/fr_fr/AWSEC2/latest/UserGuide/concepts.html`
Il importe de souligner que l'un des auteurs, A. Moradi, est *AWS Certified Solutions Architect - Associate*.

2. Brève évocation du sans-modèle [5]

2.1. *Modèle ultra-local et correcteur intelligent*

Soit un système entrée-sortie monovariable, c'est-à-dire avec une seule commande u et une seule sortie y, dont la description mathématique est inextricable. Ce fait conduit à introduire le modèle « ultra-local » :

$$y^{(\nu)} = F + \alpha u$$

où $\nu \geq 1$.

— En général, $\nu = 1$:

$$\boxed{\dot{y} = F + \alpha u} \tag{1}$$

— Le praticien choisit le paramètre constant $\alpha \in \mathbb{R}$ de sorte que les trois termes de (1) soient de magnitudes semblables. Une identification précise de α est sans objet.

— F, qui confond structure inconnue du système et perturbations externes, s'estime à chaque instant à partir de u et y.

On associe à (1) le correcteur « intelligent proportionnel », ou *iP*,

$$\boxed{u = -\frac{F_{\text{estim}} - \dot{y}_d - K_P e}{\alpha}} \tag{2}$$

— F_{estim} est une estimée F.

— y_d est la trajectoire de référence, ou consigne,

— $e = y_d - y$ est l'erreur de poursuite,

— $K_P \in \mathbb{R}$ est un gain.

Il vient, d'après (1) et (2),

$$\dot{e} + K_P e = F_{\text{estim}} - F$$

Le choix d'un K_P stabilisant est transparent. La poursuite est « bonne » si l'estimée F_{estim} l'est, c'est-à-dire si $F - F_{\text{estim}} \simeq 0$.

2.2. *Estimation de F*

Première formule

D'après une propriété classique d'analyse mathématique (voir, par exemple, [Bourbaki (1976)]), on peut, sous des hypothèses faibles, approcher F en (1) par une fonction F_{estim}, constante par morceaux. Avec les notations du calcul opérationnel (voir, par exemple, [Erdélyi (1962)]), (1) s'écrit :

$$sY = \frac{\Phi}{s} + \alpha U + y(0)$$

où Φ est une constante. On élimine la condition initiale $y(0)$ en dérivant les deux membres par $\frac{d}{ds}$:

$$Y + s\frac{dY}{ds} = -\frac{\Phi}{s^2} + \alpha\frac{dU}{ds}$$

5. Renvoyons à [Fliess & Join (2013)] pour plus de détails.

On multiplie à gauche les deux membres par s^{-2}. D'où, dans le domaine temporel, une estimée en temps réel, obtenue grâce à l'équivalence entre $\frac{d}{ds}$ et la multiplication par $-t$,

$$F_{\text{estim}}(t) = -\frac{6}{\tau^3} \int_{t-\tau}^{t} \left[(\tau - 2\sigma)y(\sigma) + \alpha\sigma(\tau - \sigma)u(\sigma)\right] d\sigma \tag{3}$$

Seconde formule

En utilisant la définition (2) de l'iP, il vient :

$$F_{\text{estim}}(t) = \frac{1}{\tau}\left[\int_{t-\tau}^{t} (\dot{y}_d - \alpha u - K_P e)\, d\sigma\right] \tag{4}$$

Remarque 2. *Quelques points utiles :*

- *Les calculs* (3) *et* (4) *se font en temps réel.*
- *Les intégrales en* (3) *et* (4) *sont des filtres passe-bas, qui atténuent le bruit.*
- *D'un point de vue pratique, l'échantillonnage mène à des filtres numériques, faciles à implanter.*

3. Mise en œuvre

On utilise le modèle ultra-local (1) et le correcteur iP (2).

- La sortie y correspond à l'utilisation des « processeurs », ou *central processing units* (*CPU*), durant le période $[t-h, t)$ d'échantillonnage :
  - Soit $\text{CPU}_i^h(t)$ la charge moyenne, sur l'intervalle $[t-h, t)$, du processeur de la machine virtuelle d'ordre i, dépendant des requêtes.
  - Soit $M_{\text{act}}^h(t)$, $M_{\min} \leqslant M_{\text{act}}^h(t) \leqslant M_{\max}$, le nombre de machines virtuelles actives.

  Alors
  $$y(t) = \sum_{i=1}^{M_{\text{act}}^h(t)} \text{CPU}_i^h(t) \tag{5}$$
- Pour la trajectoire désirée, ou consigne, y_d, on choisit
  $$y_d(t) = \frac{M_{\text{act}}^h(t)}{2} \tag{6}$$
  Ce choix est un compromis. Ainsi, une consigne égale à $0.3 \times M_{\text{act}}^h(t)$ (resp. $0.8 \times M_{\text{act}}^h(t)$) impliquerait une sous-exploitation (resp. sur-exploitation). Ajoutons que toute sur-exploitation notable induit un retard significatif dans l'exécution des requêtes. D'où une dégradation de la qualité de service.
- La commande u correspond à la cardinalité de la grappe, c'est-à-dire au nombre de machines virtuelles actives.

Remarque 3. *Certaines valeurs en ordonnées dans les figures du paragraphe* 4.2. *sont données en pourcentages de façon évidente. C'est pourquoi, par exemple,* $1/2$ *en* (6) *correspond alors à* 50%.

Le schéma bloc 1 illustre ce qui précède.

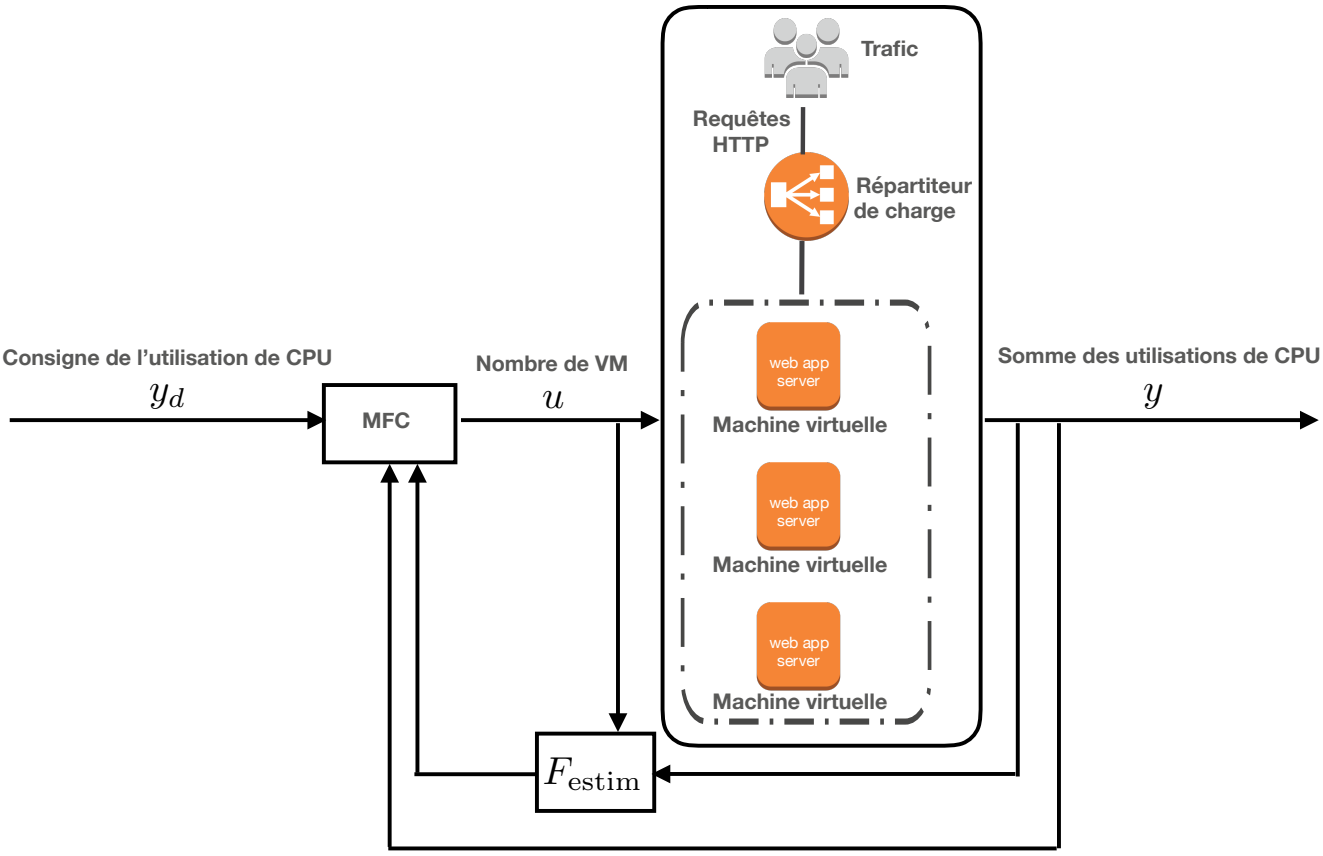

Figure 1.: Schéma de commande.

4. Expérimentations

4.1. *Cadre général*

La figure **2** représente l'infrastructure AWS.

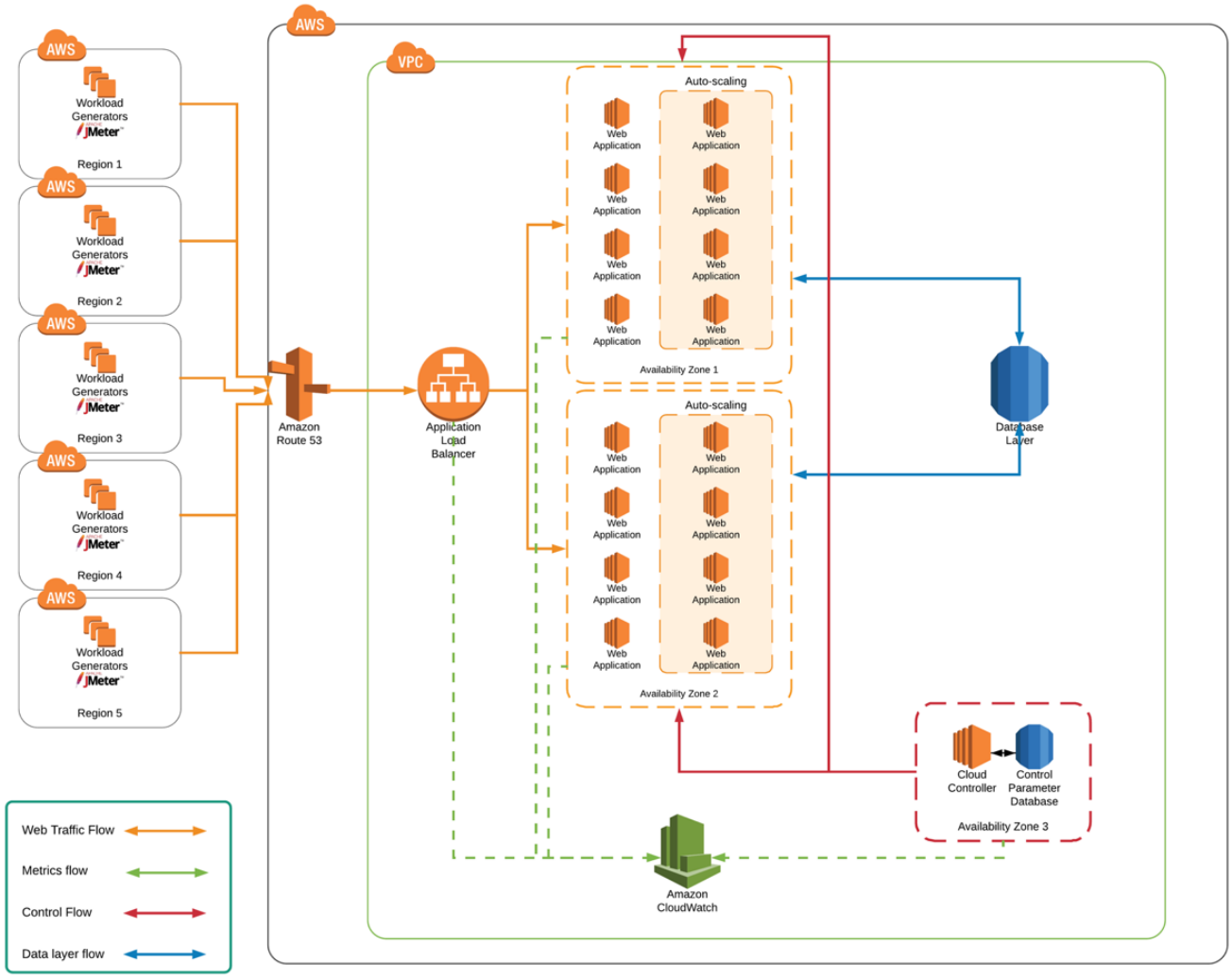

Figure 2.: Cadre expérimental.

On stocke, pour se rapprocher autant que faire se peut du fonctionnement authentique, la totalité de Wikipedia en anglais (version du 15 novembre 2017). On introduit les deux situations :

1. Un trafic en paliers, pour comparer les diverses techniques.
2. Un trafic à fortes variations, provenant du passage du trafic Wikipedia de 120 à 2 heures. En ressort l'excellente réactivité de la commande sans modèle.

On envoie dans les deux cas 1 million de requêtes pendant 2 heures.

4.2. *Expériences et comparaisons*

On expérimente d'abord le premier trafic sur une grappe AWS « statique », où le nombre de VM est constant : voir figures **3** et **4**. Quand $M^h_{\text{act}}(t) = 30$, c'est-à-dire avec 30 VM, pendant le test, la grappe est sur-dimensionnée (voir figure **3**). La charge moyenne des processeurs est inférieure à la référence (voir sous-figure *Cluster Average CPU Usage*). Avec, en revanche, $M^h_{\text{act}}(t) = 20$ (voir figure **4**), la charge reste au-dessus de la référence pendant les trois quarts de l'expérimentation, et sature. D'où beaucoup d'échecs, indiqués par la ligne orange de la sous-figure *Request Count*, et une indisponibilité du service.

On répète ce qui précède avec

1. l'ajustement AWS pour suivi de cible,
2. la commande sans modèle.

La figure **5** décrit les résultats pour la grappe avec l'algorithme *Target Tracking* d'AWS (version d'avril 2018). En fixant la référence à 50% et les autres paramètres à leurs valeurs proposées par Amazon. Cette procédure détecte le premier pic de trafic. Le nombre de VM croît alors jusqu'au maximum autorisé. Il diminue après détection d'une sous-utilisation. D'où un retard. Soulignons les faits suivants :

— l'utilisateur ne peut changer la fréquence d'échantillonnage et les seuils d'activation,

— Amazon ne détaille pas la détermination du nombre de VM.

Passons à la commande sans modèle. On observe en figure **6** l'adaptation parfaite de la grappe $M^h_{\rm act}(t)$ (sous-figure *VM count*) au nombre de requêtes. La sous-figure *TargetResponseTime* décrit un temps de réponse à peu près constant, sauf lors de la première apparition d'un pic de charge. En figure **7**, sous-figure *Reference Tracking*, on observe un suivi impeccable de l'objectif désiré. La courbe bleue représente la somme des utilisations des processeurs mesurées sur chaque VM (*cf.* (5)) et la courbe verte représente la consigne désirée (*cf.* (6)). La sous-figure *F estimated* dépeint une excellente estimation de l'allure de la charge.

Le répartiteur de charges Amazon *Elastic Load Balancing* (*ELB*), du type *Application* [6], est conçu pour s'adapter à l'arrivée des requêtes. Il présente toutefois une limitation : pendant un laps de temps de 5 minutes, le trafic ne doit pas croître de plus de 50% [7]. C'est pourquoi on introduit, ici, des variations très brutales, peu communes en pratique, pour affiner la confrontation avec le sans-modèle. D'après les figures **8** et **9**, l'iP (2) suit l'allure de la charge et adapte au mieux la taille de la grappe.

6. Voir
`https://docs.aws.amazon.com/fr_fr/elasticloadbalancing/latest/application/introduction.html`
7. Voir
`https://aws.amazon.com/articles/best-practices-in-evaluating-elastic-load-balancing`

Technique	Somme des durées de vie, en secondes, des VM utilisées	Déviation moyenne du CPU par rapport à la référence
Commande sans modèle (fig. 6)	127 920	8,53%
AWS Target Tracking (fig. 5)	187 080	21,73%
Sans élastisité, avec 20 VM (fig. 4)	144 000	28,36%
Sans élastisité, avec 30 VM (fig. 3)	216 000	21,78%

Tableau 4.1.: Comparaison des techniques.

On construit le tableau 4.1 grâce aux « indicateurs de performance », ou *Key Performance Indicators* (*KPI*), suivants :

— somme des durées de vie, en secondes, des VM utilisées pendant une expérimentation,

— déviation de mesure du capteur, c'est-à-dire de la moyenne de charge du processeur, par rapport à la référence 50% (voir les sous-figures Cluster Average CPU Usage).

Ils précisent les termes de la comparaison : l'auto-ajustement par commande sans modèle induit une forte diminution de la consommation temporelle des VM, donc du coût, tout en assurant un suivi excellent de la référence.

Remarque 4. *Les comparaisons faites avec d'autres techniques, comme les PID, semblent tout aussi favorables à notre approche. L'absence de détails dans les publications, que nous avons pu lire, nous interdit ici d'en dire plus.*

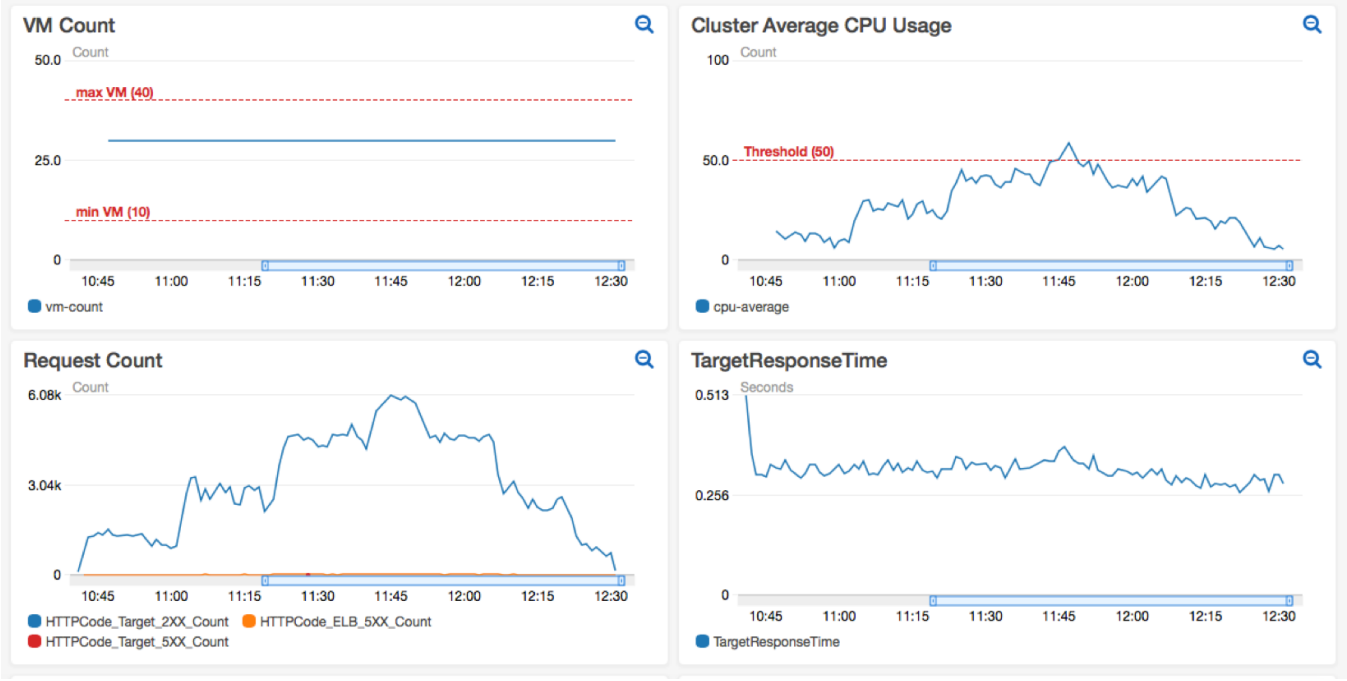

Figure 3.: Résultats expérimentaux sans auto-ajustement, avec 30 machines virtuelles.

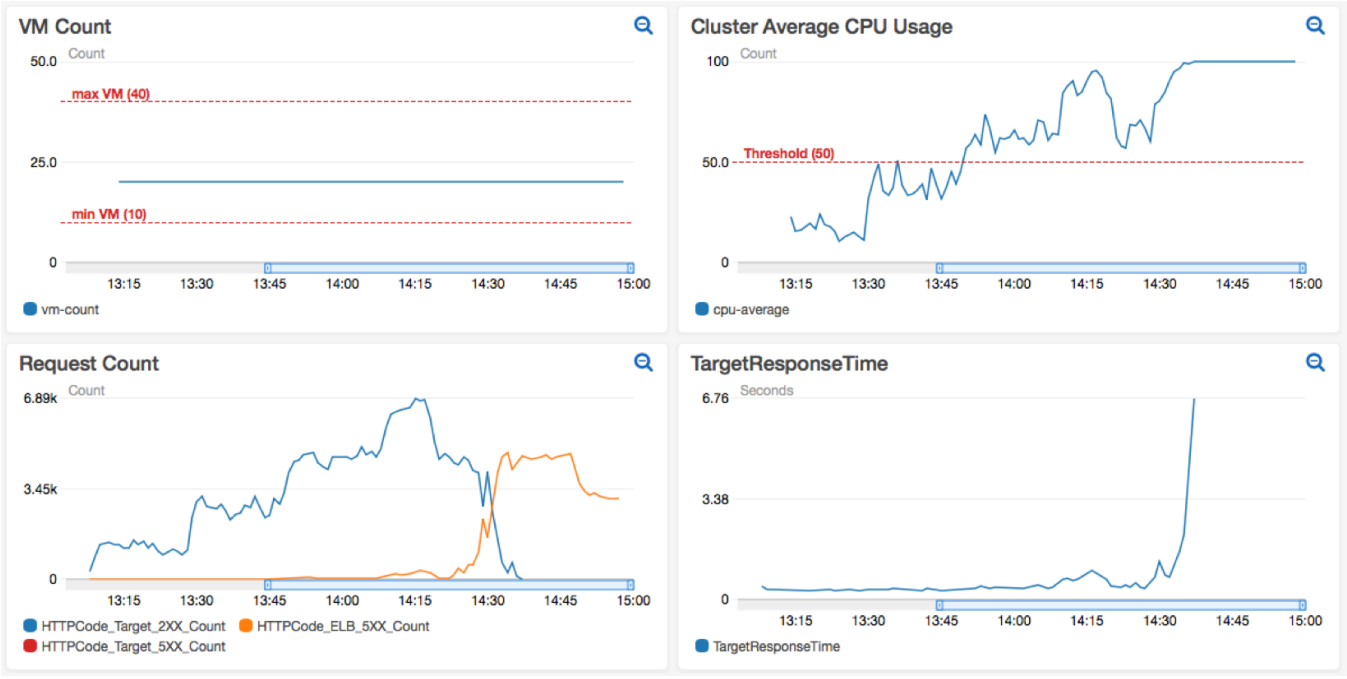

Figure 4.: Résultats expérimentaux sans auto-ajustement, avec 20 machines virtuelles.

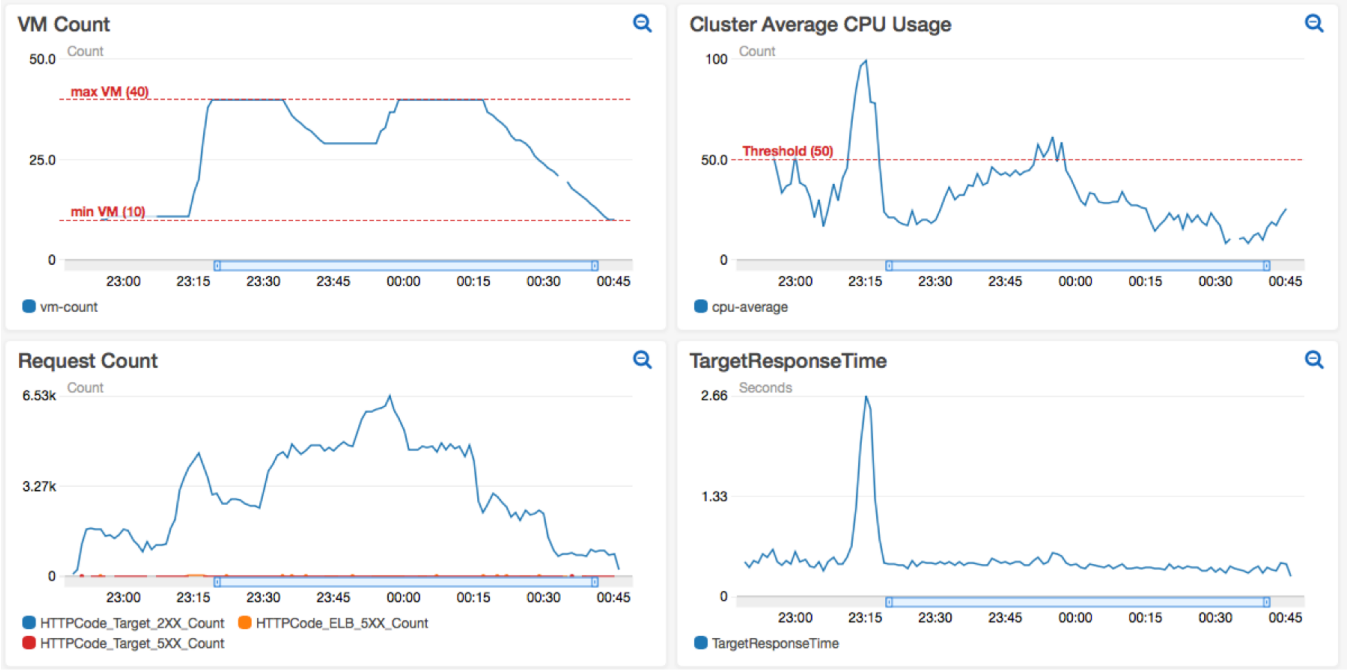

Figure 5.: Résultats expérimentaux avec *AWS Target Tracking Auto-Scaling Algorithm* et variations en **paliers** du trafic Wikipedia.

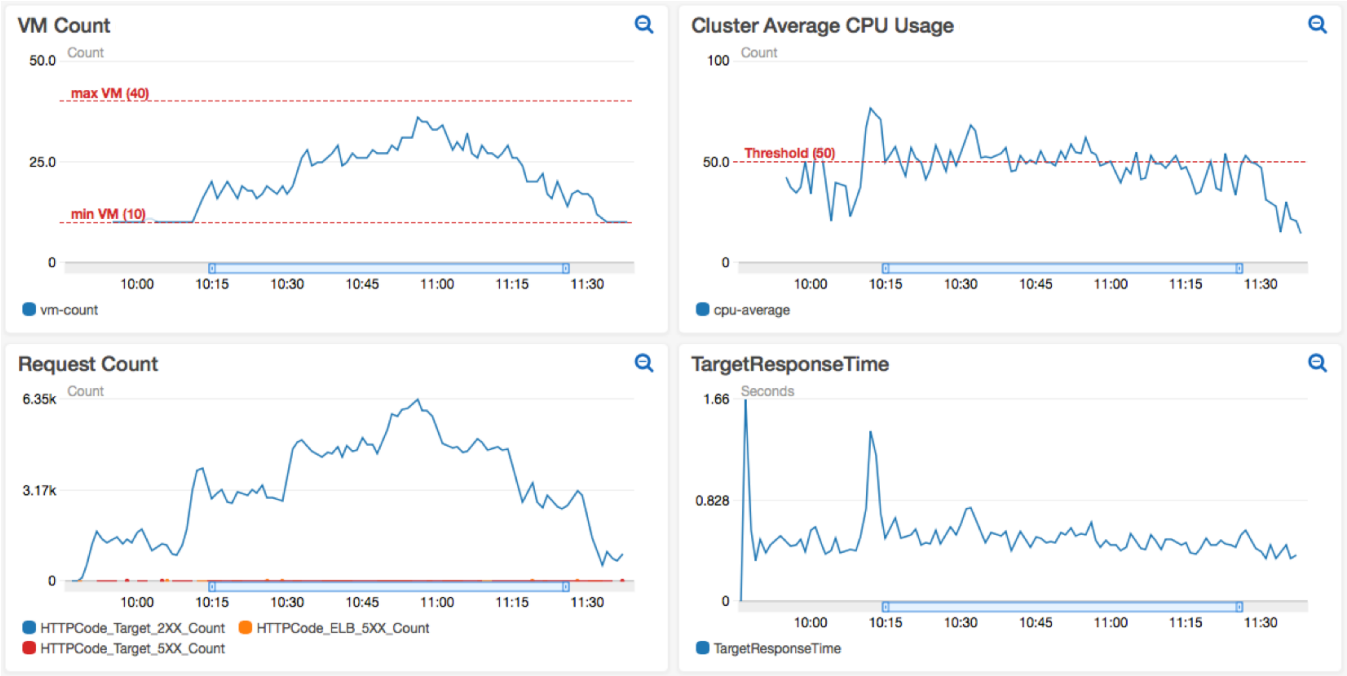

Figure 6.: Résultats expérimentaux avec commande sans modèle et variations en **paliers** du trafic Wikipedia.

5. Conclusion

La « tolérance aux pannes », ou *fault tolerance*, sujet classique en automatique (voir, par exemple, [Blanke *et coll.* (2016)]), apparaît évidemment en nuagique (voir, par exemple, [Arabnejad *et coll.* (2017)]). La défaillance de machines virtuelles y est un enjeu saillant. Comme la cardinalité des VM est la commande, on sait [Fliess & Join (2013), Lafont *et coll.* (2015)] que la commande sans modèle surmonte aisément ces aléas. De futures publications le confirmeront.

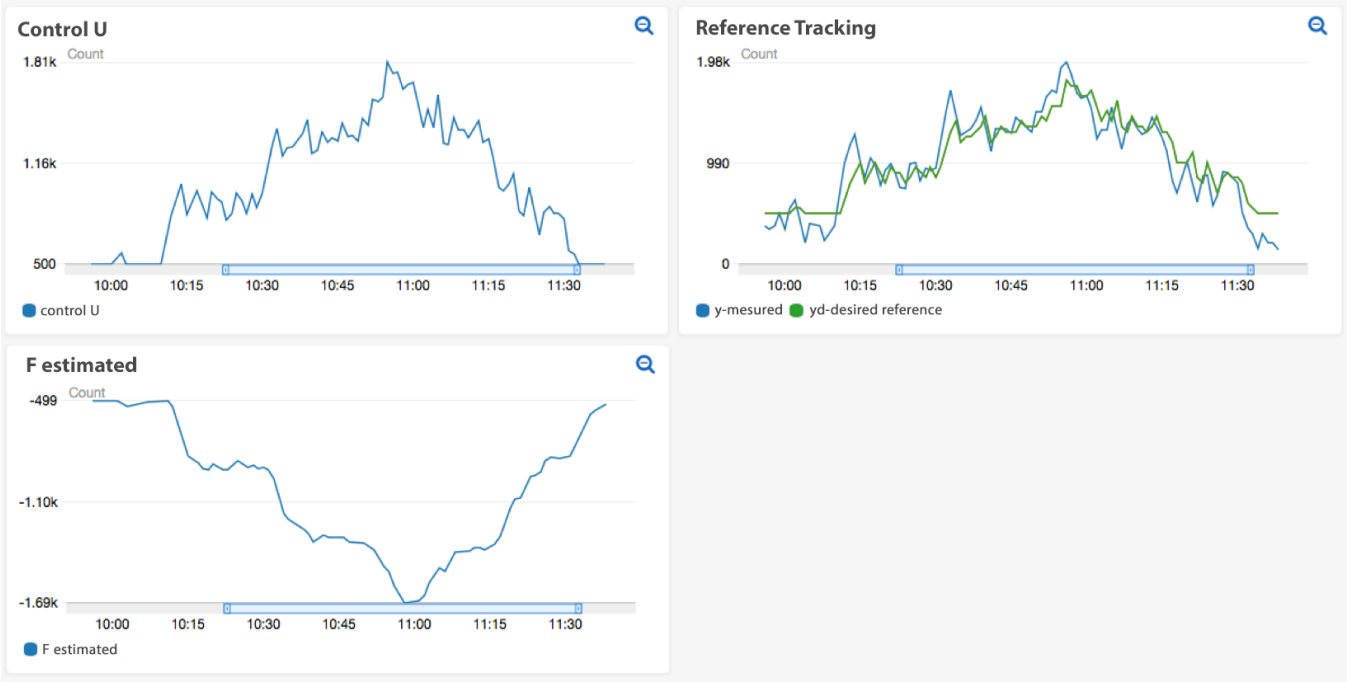

Figure 7.: Résultats avec correcteur iP expérimental ($K_p = 0.8$, $\alpha = 1$, échantillonnage : 1 min) et variations en **paliers** du trafic Wikipedia.

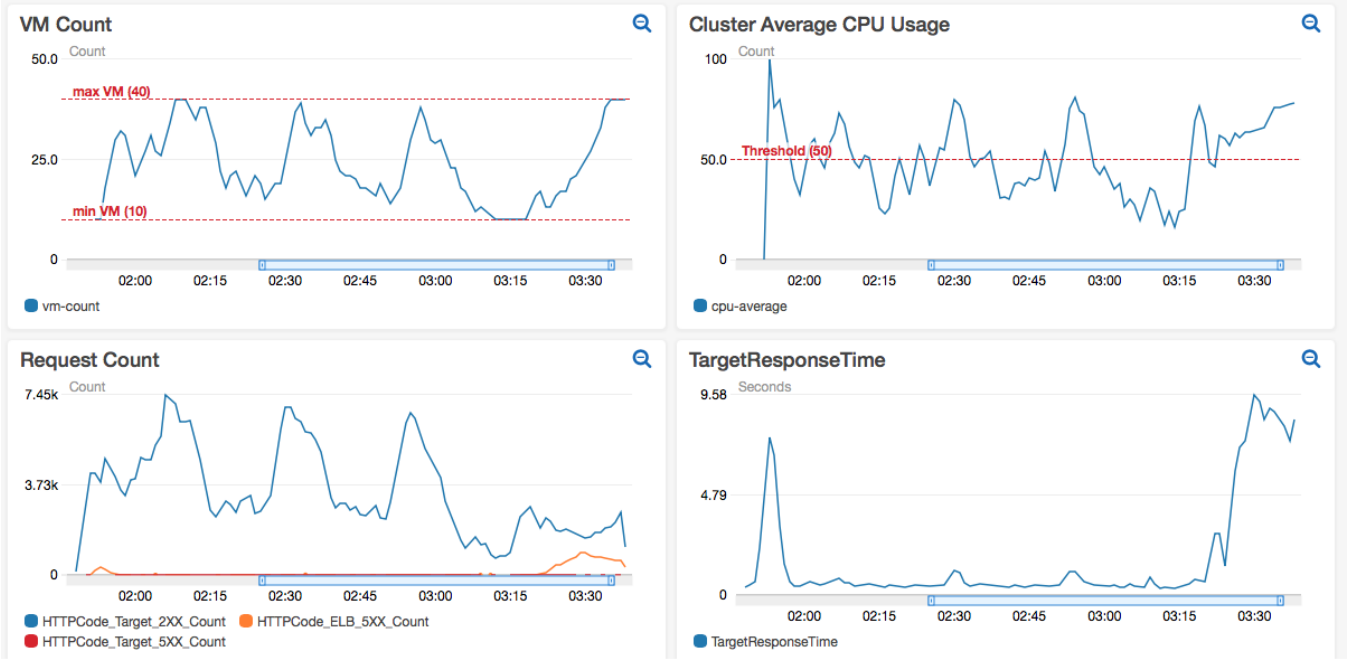

Figure 8.: Résultats expérimentaux avec commande sans modèle et variations **aigües** de trafic Wikipedia.

De prochains travaux étendront notre méthode aux « conteneurs », ou *containers*, (voir, par exemple, [Bernstein (2014), Pahl *et coll.* (2017)]) développés notamment par la compagnie Docker, de plus en plus employés. Signalons, par exemple, [Baresi *et coll.* (2016)] pour une élasticité par régulation à partir d'un modèle simplifié. Bien d'autres thèmes, assez voisins de ce travail, méritent également considération. Mentionnons-en un. Une démarche analogue devrait permettre d'aborder les nœuds de calcul, comme *MapReduce*, dû à Google. On les utilise pour l'analyse des « mégadonnées », ou *big data*. Les requêtes précédentes deviennent celles des décideurs à propos des « jeux de données », ou *datasets*

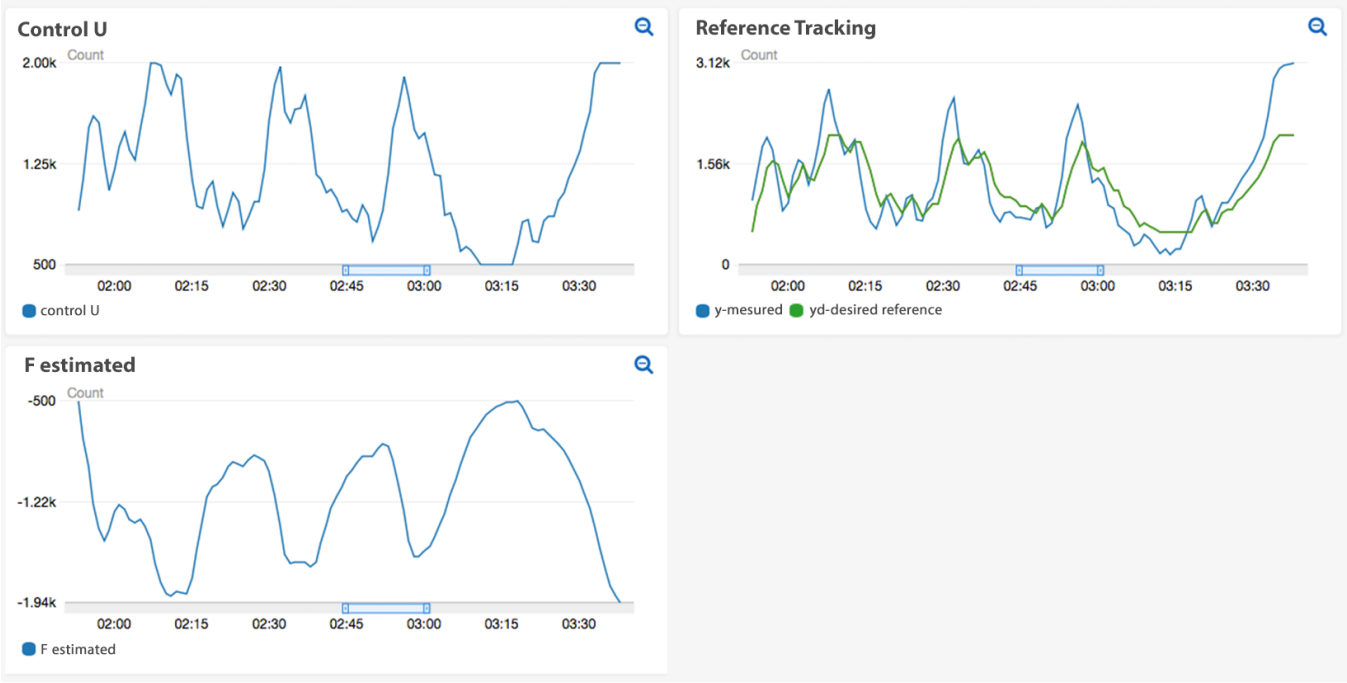

Figure 9.: Résultats avec correcteur iP expérimental ($K_p = 0.8$, $\alpha = 1$, échantillonnage : 1 min) et variations **aigües** du trafic Wikipedia pour la grappe AWS.

[Berekmeri *et coll.* (2016), Cerf *et coll.* (2016), Cerf *et coll.* (2017)].

Si « les techniques sont des procédés bien définis et transmissibles destinés à produire certains résultats jugés utiles » [Lalande (1926)], le nuagique est, alors, une collection de techniques plutôt qu'une science. Le contenu de cet article devrait, s'il se confirme, infléchir cet état, du moins partiellement. Ce serait, aussi, le signal, longtemps pressenti, de la place éminente qui revient à l'automatique en informatique, surtout si l'on abandonne l'ambition, trop souvent vaine, pour ne pas dire naïve, d'une modélisation mathématique précise, qu'elle soit déterministe ou probabiliste. Que les mathématiciens se rassurent ! Leur rôle ne sera en rien diminué s'ils participent à l'édification des nouveaux outils exigés, dont la commande sans modèle n'est qu'un exemple parmi bien d'autres à développer [8] sinon à inventer.

Bibliographie

AL-DHURAIBI Y., PARAISO F., DJARALLAH N., MERLE P., « Elasticity in cloud computing: State of the art and research challenges ». *IEEE Transactions on Services Computing*, 11 (2018) : 430-447.

ALKHARIF S., LEE K., KIM H., « Time-series analysis for price prediction of opportunistic cloud computing resources ». W. Lee, W. Choi, S. Jung, M. Song (Eds.) : *Proceedings of the 7th International Conference on Emerging Databases*, Lecture Notes in Electrical Engineering 461, pp. 221-240, Singapour : Springer, 2018.

ARABNEJAD H., PAHL C., ESTRADA G., SAMIR A., FOWLEY F., « A fuzzy load balancer for adaptive fault tolerance management in cloud platforms ». F. De Paoli, S. Schulte, E.B. Johnsen (Eds.) : *Service-Oriented and Cloud Computing*, Lecture Notes in Computer Science 10465, pp. 109-124, Cham, Suisse : Springer, 2017.

ARMBRUST M., FOX A., GRIFFITH R., JOSEPH A.D., KATZ R., KONWINSKI A., LEE G., PATTERSON D., RABKIN A., STOICA I., ZAHARIA M., « A view on cloud computing ». *Communications of the ACM*, 53 (2010) : 50-58.

8. Voir, par exemple, [Fliess *et coll.* (2018)] sur les « chroniques », ou *time series* : « mégadonnées », ou *big data*, et « apprentissage », ou *machine learning*, y jouent un rôle moindre qu'ailleurs. Il convient de mentionner que des points de vue classiques sur les chroniques ont déjà été employés assez souvent en nuagique (voir, par exemple, [Alkharif *et coll.* (2018), Calheiros *et coll.* (2015)] et leur bibliographie).

ÅSTRÖM K.J., HÄGGLUND T., *Advanced PID Control*. Research Triangle Park, NJ : Instrument Society of America, 2006.

ÅSTRÖM K.J., MURRAY R.M., *Feedback Systems*: *An Introduction for Scientists and Engineers*. Prinveton, NJ : Princeton University Press, 2008.

BARA O., FLIESS M., JOIN C., DAY J., DJOUADI S.M., « Toward a model-free feedback control synthesis for treating acute inflammation ». *Journal of Theoretical Biology*, 448 (2018) : 26-37.

BARESI L., GUINEA S., LEVA A., QUATTROCCHI G., « A discrete-time feedback controller for containerized cloud applications ». *ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, Seattle, 2016.

BEREKMERI M., SERRANO D., BOUCHENAK S., MARCHAND N., ROBU B., « Feedback autonomic provisioning for guaranteeing performance in MapReduce systems ». *IEEE Transactions on Cloud Computing*, (2016) http://doi.ieeecomputersociety.org/10.1109/TCC.2016.2550047

BERNSTEIN D., « Containers and cloud: From LXC to Docker to Kubernetes », *IEEE Cloud Computing*, 1 (2014) : 81-84.

BERSTEL J., REUTENAUER C., *Noncommutative Rational Series with Applications*. Cambridge : Cambridge University Press, 2010.

BLANKE M., KINNAERT M., LUNZE J., STAROSWIECKI M., *Diagnosis and Fault-Tolerant Control* (3rd ed.). Heidelberg : Springer, 2016.

BOURBAKI N., *Fonctions d'une variable réelle*. Paris : Hermann, 1976.

BU X., RAO J., XU C.Z., « Coordinated self-configuration of virtual machines and appliances using a model-free learning approach ». *IEEE Transactions on Parallel and Distributed Systems*, 24 (2013) : 681-690.

BUYYA R., VECCHIOLA C., SELVI S.T., *Mastering Cloud Computing*: *Foundations and Applications Programming*. Waltham, MA: Morgan Kaufmann, 2013.

CALHEIROS R.N., MASOUMI E., RANJAN R., BUYYA R., « Workload prediction using ARIMA model and its impact on cloud applications' QoS ». *IEEE Transactions on Cloud Computing*, 3 (2015) : 449-458.

CERF S., BEREKMERI M., ROBU B., MARCHAND N., BOUCHENAK S., « Cost function based event triggered model predictive controllers – Application to big data cloud services ». *55th IEEE Conference on Decision and Control (CDC)*, Las Vegas, 2016.

CERF S., BEREKMERI M., ROBU B., MARCHAND N., BOUCHENAK S., LANDAU I.D., « Adaptive feedforward and feedback control for cloud services ». *IFAC PapersOnLine* 50-1 (2017) : 5504-5509.

CHONG E.K.P., « The control problem ». *IEEE Control Systems*, 37 (2017) : 14-16.

EILENBERG S., *Automata, Languages and Machines* (vol. A). New York : Academic Press, 1974.

ERDÉLYI A., *Operational Calculus and Generalized Functions*. New York : Holt Rinehart Winston, 1962.

FEHLING C., LEYMANN F., RETTER R., SCHUPECK W., ARBITTER P., *Cloud Computing Patterns – Fundamentals to Design, Build, and Manage Cloud Applications*. Vienne : Springer, 2014.

FLIESS M., « Un codage non commutatif pour certains systèmes échantillonnés non linéaires ». *Information and Control*, 38 (1978) : 264-287.

FLIESS M., JOIN C., « Model-free control ». *International Journal of Control*, 86 (2013) : 2228-2252.

FLIESS M., JOIN C., « Deux améliorations concurrentes des PID ». *ISTE OpenScience Automatique*, 2 (2018) : 23 p. https://hal.archives-ouvertes.fr/hal-01687952/en/

FLIESS M., JOIN C., VOYANT C., « Prediction bands for solar energy: New short-term time series forecasting techniques », *Solar Energy*, 166 (2018) : 519-528

GALANTE G., DE BONA L.C.E., « A survey on cloud computing elasticity ». *IEEE 5th International Conference on Utility and Cloud Computing (UCC)*, Chicago, 2012.

HELLERSTEIN J.L., DIAO Y., PAREKH S., TILBURY D.M., *Feedback Control of Computing Systems*. Hoboken, NJ : Wiley, 2004.

HERBST N., KOUNEV S., REUSSNER R., « Elasticity in cloud computing: What it is, and what it is not ». *10th International Conference on Autonomic Computing (ICAC)*, San Jose, CA, 2013.

JANERT P.K., *Feedback Control for Computer Systems*. Sebastopol, CA : O'Reilly Media, 2014.

JOIN C., CHAXEL F., FLIESS M., « "Intelligent" controllers on cheap and small programmable devices ». *2nd International Conference on Control and Fault-Tolerant Systems (SysTol'13)*, Nice, 2013. https://hal.archives-ouvertes.fr/hal-00845795/en/

KALMAN R. E., FALB P. L., ARBIB M. A., *Topics in Mathematical System Theory*, New York : McGraw-Hill, 1969.

LAFONT F., BALMAT J.-F., PESSEL N., FLIESS M., 2015. « A model-free control strategy for an experimental greenhouse with an application to fault accommodation ». *Computers and Electronics in Agriculture*, 110 (2015) : 139-149.

LALANDE A., *Vocabulaire technique et critique de la philosophie*. Paris : Alcan, 1926 (Réédition, Paris : PUF, 2010).

LEVA A., MAGGIO M., PAPDOPOULOS A.V., TERRANEO F., *Control-Based Operating System Design*. Londres : The Institution of Engineering and Technology, 2013.

LORIDO-BOTRAN T., MIGUEL-ALONSO J., LOZANO J.A., « A review of auto-scaling techniques for elastic applications in cloud environments ». *Journal of Grid Computing*, 12 (2014) : 559-592.

MARINESCU D.C., *Cloud Computing*: *Theory and Practice* (2nd ed.). Cambridge, MA : Morgan Kaufmann, 2017.

O'DWYER A., *Handbook of PI and PID Controller Tuning Rules* (3rd ed.). Londres : Imperial College Press, 2009.

PAHL C., BROGI A., SOLDANI J., JAMSHIDI P., « Cloud container technologies: a state-of-the-art review ». *IEEE Transactions on Cloud Computing*, 2017. doi:10.1109/TCC.2017.2702586

PATIKIRIKORALA T., COLMAN A., HAN J., WANG L., « A systematic survey on the design of self-adaptive software systems using control engineering approaches ». *7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, Zürich, 2012.

RAO J., WEI Y., GONG J., RAO J., XU C.-Z., « DynaQoS: Model-free self-tuning fuzzy control of virtualized resources for QoS provisioning ». *19th IEEE International Workshop on Quality of Service*, San Jose, 2011.

RIVARD F., *Cloud Computing : Le système d'information sans limite*. Paris : Hermes-Lavoisier, 2012.

SAKAROVITCH J., *Éléments de théorie des automates*. Paris : Vuibert, 2003.Traduction anglaise : *Elements of Automata Theory*. Cambridge : Cambridge University Press, 2009.

SCHÜTZENBERGER M.P., « On the definition of a family of automata ». *Information and Control*, 4 (1961) : 245-270.

SONTAG E.D., *Mathematical Control Theory* (2nd ed.). New York : Springer, 1998.

ULLAH A., LI J., SHEN Y., HUSSAIN A., « A control theoretical view of cloud elasticity: taxonomy, survey and challenges ». *Cluster Computing* (2018) https://doi.org/10.1007/s10586-018-2807-6

VICAT-BLANC PRIMET P., SOUDAN S., GUILLIER R., GOGLIN B., *Réseaux de calcul : Des grappes aux nuages de calcul*. Paris : Hermes-Lavoisier, 2010.

WANG X., DU Z., CHEN Y., « An adaptive model-free resource and power management approach for multi-tier cloud environments », *Journal of Systems and Software*, 85 (2012) :1135-1146.

WITTIG M., WITTIG A., *Amazon Web Services in Action*. Shelter Island, NY : Manning, 2016.

WU C., BUYYA R., *Cloud Data Centers and Cost Modeling*. Waltham, MA : Morgan Kaufmann, 2015.

