Compas'2022 : Parallélisme / Architecture/ Système
MIS/UPJV - Amiens France, 5-8 juillet 2022

Adirbs : Approche de Dimensionnement des Infrastructures de Robotique en Brouillard basée sur la Simulation

Lucien Arnaud Ndjie Ngale * et Mélanie Fontaine †

ENS de Lyon,
Laboratoire LIP - 46 allée d'Italie
69007 Lyon - France
lucien.ndjie@ens-lyon.fr

Laboratoire des Technologies Innovantes,
48 Rue d'Ostende
Saint Quentin - France
melanie.fontaine@crispi-upjv.fr

Résumé
De nombreux projets de robotique sont confrontés soit aux limites financières de la gestion locale des données, soit à la latence manifeste d'une délocalisation complète de la gestion des données. Pour résoudre ce problème, les infrastructures Fog ont été créées, rapprochant le calcul distant des sources de données. Dans le cas où ces sources de données sont des robots, nous parlons de Plateformes de Fog Robotique (PFR). Néanmoins, dimensionner de telles plateformes, afin de minimiser leurs besoins en ressources Cloud, reste un défi critique. Cet article présente une Approche de Dimensionnement des Infrastructures de Robotique en Brouillard basée sur la Simulation (Adirbs). Nous l'appliquons à un cas d'utilisation dans le secteur médicale. Les résultats obtenus, nous permettent de déterminer rapidement le nombre efficace de ressources à louer, leurs spécifications (puissance de calcul, capacités de stockage et mémoires) pour des Clouds de type IaaS ainsi que la meilleure politique d'ordonnancement des requêtes sur ces ressources.

Mots-clés : Fog computing, Robotique, Cloud computing, Simulation, Systèmes distribués.

1. Introduction

Aujourd'hui, l'utilisation de robots (l'interaction homme-robot) est plus que jamais d'actualité, car elle facilite l'automatisation des tâches et garantit une meilleure précision et rapidité d'exécution par rapport au traitement humain [4, 12, 15]. Cependant, les propriétaires d'équipements robotiques sont souvent confrontés à des problèmes liés à la gestion locale des données massives [18] générées par ces robots et, parfois, obligés de prendre en charge la mise à jour de leurs fonctionnalités lorsque les spécificités de l'écosystème changent. Trois types de solutions sont alors disponibles : (a) une utilisation de robots puissants en calcul et en stockage (gestion entièrement locale des données, qui multiplie, les robots et les ressources lorsque les spécificités de l'écosystème changent), (b) une utilisation exclusive des ressources distantes (avec une

*. Sous la supervision d'Eddy Caron (ENS de Lyon) et Huaxi (Yulin) Zhang (UPJV)
†. Sous la supervision de Huaxi (Yulin) Zhang (UPJV)

latence généralement importante, du fait que les requêtes soient répétitives et nombreuses) et (c) une utilisation hybride des ressources (faibles en local mais pouvant être complétées à distance). Cette dernière approche est un bon compromis entre les solutions (a) et (b). Dans ce dernier cas de figure, lorsque les sources de données sont des équipements robotiques et que les ressources distantes sont des ressources Cloud, on parle de Plateformes de Fog Robotique (PFR) [7]. Ces PFR sont caractérisées par : (i) deux couches matérielles (Edge et Cloud), constituées de ressources [3] ; (ii) un ensemble de flux de travail qui modélisent des façons indépendantes d'utiliser la PFR ; (iii) la spécificité des requêtes qui sont (périodiques, indépendantes et répétitives). Comment dimensionner et faire passer à l'échelle de telles plateformes ? Un grand nombre de travaux traitent principalement le déploiement et le dimensionnement des PFR en implémentant le concept de Robot as a Service (RaaS [9]) qui fournit un contrôle distant des robots via des applications en ligne. Cependant, cette approche se concentre uniquement sur l'optimisation des communications entre robots afin d'assurer une meilleure qualité de service [16] aux utilisateurs finaux. Néanmoins, ceci reste limité en ce qui concerne la minimisation des besoins en ressources Cloud des robots. Comment évaluer dans ce cas, ces besoins ? Et comment distribuer efficacement les requêtes de la couche Edge sur ces ressources ?
Cet article présente une nouvelle approche (**Adirbs**) pour déterminer conjointement les besoins minimaux en ressources et une politique d'ordonnancement des requêtes sur ces ressources. Basée sur des simulations de déploiement de PFR, elle vise à réduire la durée d'activité sur une PFR ainsi que le coût financier d'utilisation des ressources de manière efficace. Elle permet d'obtenir une politique d'ordonnancement des ressources qui minimise : le makespan (temps de complétion des requêtes), les degrés de non équilibrage de charges et des stockages et le nombre de ressources, par le biais d'un algorithme (pour des solutions IaaS) pour une complexité maximale de $O(\log^2(nbq) \times C_{sim}(nbq))$ ($C_{sim}(nbq)$ étant la pire complexité d'une simulation de l'émission de nbq requêtes).

2. Travaux relatifs

De nombreux travaux présentent le déploiement d'écosystèmes robotiques sur des systèmes distribués à grande échelle dans différents paradigmes comme SOA, Cloud computing, IoT ou P2P : **(a) SOA et Cloud :** RobotWeb [10] est un système intergiciel de services web basé sur SOAP qui virtualise, expose les ressources informatiques des robots en tant que services et les publie à l'intention des utilisateurs finaux. Dans [10], les robots fonctionnent avec le système ROS [11], car il fournit une abstraction matérielle qui facilite le développement d'applications ; **(b) Edge Computing :** Le travail présenté dans [21] propose un nouveau concept qui répond aux problèmes de soutien des activités de contrôle et de surveillance sur les sites de déploiement et d'automatisation industrielle. L'objectif est d'agir pour contrôler ou diffuser des objets robotiques statiques ou dynamiques qui sont conscients de leur position dans le monde physique de manière transparente en fournissant un moyen de les faire fonctionner comme le suggère le concept d'Internet des objets robotiques (IoRT) [21] .
Dans le RaaS, tout est mis en œuvre pour que l'utilisateur final n'ait pas à se soucier de la mise à jour de la fonctionnalité. Ce paradigme d'informatique oriente naturellement les travaux liés à la Fog Robotique vers l'optimisation des communications robot, donc vers celle du ROS (Robot Operating System) [20]. En ce sens, [16] lie la bibliothèque d'optimisation NLopt [8] à ROS. Grâce à cette bibliothèque, ils optimisent l'infrastructure de communication des scénarios robotiques, favorisant ainsi la gestion des ressources limitées dans divers autres environnements

3. mémoire, une unité de traitement et une unité de stockage

IoT mobiles.
Il existe également, de nombreux travaux qui considèrent l'amélioration de la qualité de service du point de vue de l'utilisation des ressources sur les plateformes de Cloud en général (ordonnancement des tâches sur ces ressources [5]) : [19] étudie les algorithmes d'ordonnancement des tâches basés sur la qualité de service. L'objectif est d'améliorer l'efficacité de ces algorithmes en considérant le temps d'exécution d'une tâche sur une ressource et le coût d'utilisation des communications ; [17] propose EPOS Fog, qui introduit un système multi-agent décentralisé pour l'apprentissage collectif qui utilise les nœuds edge-to-cloud pour équilibrer conjointement la charge de travail d'entrée à travers le réseau et minimiser le coût impliqué dans l'exécution du service.
Ce travail traite principalement des besoins en ressources des dispositifs Fog robotiques de la couche Edge, puisque ce type de déploiement est un bon compromis pour la majorité des critères de comparaison (bonne latence, meilleure sécurité, bonne gestion des données). Nous nous distinguons des travaux existants en ce sens que nous évaluons la QoS du point de vue de l'utilisation des ressources en nuage, en nous basant simultanément sur plusieurs métriques telles que le makespan, le taux de traitement des requêtes, l'équilibrage de la charge, l'équilibrage du stockage et le coût d'utilisation des ressources. Nous nous inspirons des travaux sur les algorithmes d'ordonnancement les plus efficaces et les plus couramment utilisés dans les environnements en Cloud [14] : FIFO, MET, Max-Min, Round-Robin.

3. L'approche Adirbs

3.1. Modélisation mathématique

Le modèle logiciel de Adirbs (Figure 1), montre que l'exécution du workflow génère des requêtes caractérisées, chacune par : une date de début, un temps de traitement, qui est aussi son délai, et une quantité de données communiquées ; Notons $R = \{(b_i, t_i, d_i)\}_{i=1}^{q}$ un ensemble de q requêtes émises. Ce modèle exprime aussi le fait que la couche Cloud de la plateforme soit constituée de ressources, caractérisées par une puissance de calcul (en flops), une charge de travail (en FLOPs), une capacité de stockage, une mémoire, sa date de début et un makespan individuel ; Notons $S = \{(pc_j, cf_j, as_j, r_j, bd_j, ms_j)\}_{j=1}^{s}$ un ensemble de s ressources Cloud. il apparaît en plus les besoins globaux en ressources des équipements du fournisseur d'accès aux fonctionnalités du robot : la charge de travail et la quantité de stockage globales respectivement **af** et **as**. Notons alors s_{min}, et δ_{ij} respectivement, le nombre de ressources recherchées et une variable indiquant si la requête i est traitée par la ressource j ; ce qui nous donne les relations suivantes :

$$\delta_{ij} = \begin{cases} 1 & \text{si la requête } i \text{ est traitée par ressource } j \\ 0 & \text{sinon.} \end{cases} \quad (1)$$

$$pc_j = \frac{cf_j \times af}{ms_j \times \sum_{k=1}^{s_{min}} cf_k} \quad (2)$$

$$as_j = \frac{\sum_{i=1}^{q} \delta_{ij} \times d_i \times as}{\sum_{i=1}^{q} d_i} \quad (3)$$

En particulier, s'il existe plusieurs types de ressources dédiées (instances[4] de machines virtuelles du Cloud), N_{l_min} qui représente le nombre effectif d'instances peut être calculé à partir des caractéristiques : mémoire, capacité de stockage, capacité de calcul, prix par unité de temps. Notons $S' = \{(r_l, as_l, pc_l, poi_l)\}_{l=1}^{p}$, l'ensemble des types d'instances. Le modèle Adirbs vise à réduire le makespan de l'activité du Cloud et également le coût d'utilisation des instances (équation 5). Comme ces métriques évoluent de manière inverse, Adirbs utilise une métrique

4. Une instance est un ensemble de plusieurs ressources.

de pourcentage de perte (équation 4, où val, opt_val représentent respectivement les valeurs actuelles et optimales de la métrique évaluée) par rapport aux situations optimales pour chacune d'entre elles. Adirbs calcule pour chaque scénario de plate-forme, les pourcentages de perte respectivement par rapport aux situations où le makespan et le coût sont minimaux. Il calcule ensuite l'écart type de ces pourcentages pour chaque scénario et tente de le minimiser.

$$loss = \frac{|val - opt\_val|}{opt\_val} \tag{4}$$

$$cost_l = \sum_{j=1}^{n_l} ms_j \times poi_l \tag{5}$$

Où n_l est le nombre d'instances de type l, et ms_j est le makespan de la j ème instance de type l.

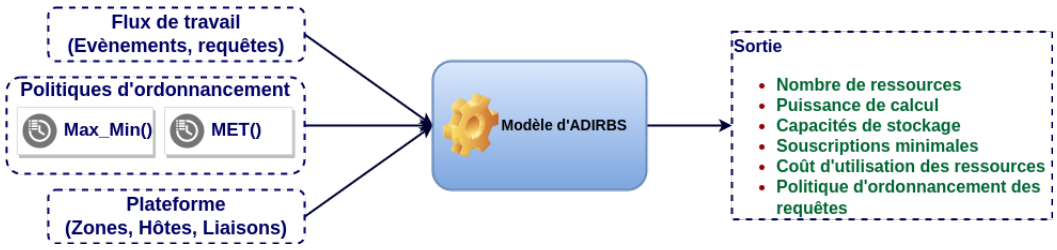

FIGURE 1 – Le Modèle Logiciel d'Adirbs.

3.2. Les étapes de Adirbs

La méthodologie de base pour l'analyse des plateformes cloud, propose quatre phases principales : (1) identification des workflows du cas d'utilisation (identifier les différentes façons dont la PFR peut être utilisé), (2) modélisation des interactions entre les dispositifs dans chaque workflow, (3) simulation de ces workflows, et (4) extraction des métriques d'évaluation et analyse des résultats. Adirbs particularise ensuite les deux dernières phases en 3 étapes.

Etape 1 - Détermination des besoins en ressources du workflow : Effectuer une simulation initiale avec n'importe quelle politique d'ordonnancement et une seule ressource Cloud, afin de déterminer la mémoire de chaque ressources (quantité maximale de données communiquée par le nombre maximal de requêtes en file d'attente), la capacité de stockage et la charge de travail totales ;

Étape 2 - Calcul du nombre d'instances satisfaisant les besoins en ressources du workflow (mémoire, stockage et charge de travail) : (1) Pour chaque type d'instances dont la mémoire est supérieure à la mémoire, on calcule le nombre minimal d'instances qui satisfont le stockage et la charge de travail totale. En outre, l'algorithme dont l'écart-type des charges de travail est minimal est sélectionné. (2) On choisit le type d'instances et la politique d'ordonnancement qui minimise le coût d'utilisation des instances.

Étape 3 - Détermination du nombre effectif d'instances (N_{l_min}) : En considérant l l'ordre du type d'instance choisi à l'étape 2, et en notant N_l le nombre d'instances satisfaisant les contraintes du workflow, pour chaque valeur de n prise dans $\{N_l \times 2^k\}_{k=0,m}$, de sorte que $N_l \times 2^m \leq nbr\_max$ (nbr_max est le nombre maximum d'instances) et, pour la politique d'ordonnancement sélectionnée à l'étape 2, simuler l'exécution des workflows sur la PFR constituée de n instances. A la fin de cette étape, toutes les traces (makespans des ressources, FLOPs calculés,

quantité de données) des simulations sont sauvegardées. Ensuite, la valeur du coude du graphique donnant l'écart-type des pourcentages de perte (associés respectivement au makespan et au coût) en fonction du nombre d'instances, est la valeur du nombre minimum d'instances requises. Adirbs propose une optimisation basée sur un algorithme, en $O(\log^2(q) \times C_{sim}(q))$.

4. Cas d'étude : l'architecture fonctionnelle d'AiBle

On s'intéresse ici à une solution qui vise à fournir un exosquelette intelligent utilisant les ressources du Cloud, pour la réhabilitation des membres supérieurs du corps humain. Ce cas d'étude est lié au projet européen (AiBle [5]). L'architecture fonctionnelle de ce projet, est constituée de deux couches et de trois zones : une couche Edge qui entretient des communications de bande passante 100 MBps (excepté celle entre l'EMG [6] et le nœud de calcul périphérique, qui vaut 640 MBps) et de latence 5 ms, ainsi qu'une couche Cloud. La couche Edge est représentée par deux zones, celle du patient (un exosquelette, un poste de travail pour le thérapeute, un EMG, un nœud de calcul périphérique et caméra Kinect V1) et celle du médecin distant (post de travail du médecin). La couche Cloud quant à elle est constituée d'une zone d'infrastructure matérielle (ensemble des ressources). Pendant une durée de 15 minutes, l'EMG communique les paramètres du patient (avec une fréquence de 1000 requêtes par seconde, en raison de 104 octets par requêtes) au Cloud via son nœud de calcul local et le poste de travail du thérapeute. Celui-ci peut contrôler si nécessaire la trajectoire de l'exosquelette et renvoie des informations de changement au Cloud. Le poste de travail du thérapeute est équipé d'une caméra qui capture les mouvements et les positions de l'exosquelette, les traite pour l'affichage et renvoie les données (en raison de 20275200 Bps) au Cloud.

5. Expérimentations

Dans cette section, nous évaluons : l'équilibre de la charge, l'équilibre du stockage et le pourcentage de perte en fonction du nombre de ressources. Nous dérivons la meilleure politique d'ordonnancement à partir de l'évaluation de l'équilibrage de la charge et du stockage, tandis que les besoins en ressources sont déterminés par le pourcentage de perte relatif au makespan et au coût. Nous comparons cinq types d'instance d'OVHCloud [7] : d2-2, b2-7, c2-30, r2-60 et t2-180. Nous sélectionnons trois politiques d'ordonnancement efficaces sur les environnements Cloud et pour des requêtes périodiques (MET(Minimum Execution Time) : complète toujours en premier lieu la tâche dont le temps d'exécution total est le plus court, Max-Min [2, 6] et FIFO [22, 13]). Chacun d'entre eux a été combiné avec Round-Robin [14], qui est l'un des meilleurs algorithmes d'équilibrage de charge dans les environnements Cloud. Nous utilisons le framewrok SimGrid [3] pour la simulation [8], car il démontre de meilleurs qualificatifs par rapport à de nombreux critères [1] par rapport à d'autre simulateurs de ce type. De plus, il est soutenu par une large communauté et fait l'objet de centaines de projets. Le tableau 1 présente les résultats expérimentaux pour trois configurations d'instances, qui satisfont les besoins en ressources du flux de travail : (a) celle qui minimise le coût : cette configuration donne le nombre minimum d'instances ainsi que l'instance qui minimise le coût d'utilisation du PFR; Cependant, elle ne minimise pas nécessairement le makespan, car en effet, nous observons un makespan manifeste pour cette configuration (e.g. : 24070 h pour AiBle), (b) celle qui minimise

5. AiBle : An efficient upper limbs réhabilitation exoskelton, https ://www.euaible.com/fr/
6. Electromyogramme
7. Cloud pricing : Comparaison des offres de cloud public - OVH, https ://www.ovhcloud.com/fr/public-cloud/prices/
8. Son code est disponible ici : **https ://gitlab.inria.fr/lndjieng/frissa**

le makespan : cette configuration correspond au nombre maximum de ressources fixé, et maximise le coût d'utilisation de la PFR (61298 euros pour AiBle). (c) Une configuration hybride qui minimise le pourcentage de pertes, qui tente de faire un compromis entre les deux premières. Elle est 46.09% plus chère que la première et 33.62% plus lente que la seconde. Le tableau 1 montre également que, les petites machines ont tendance à être sélectionnées par Adirbs car le nombre de petites machines qui équivalent à une plus grande machine revient moins cher que d'avoir une seule instance de la plus grande machine. En outre, le nombre d'instances qui satisfont le stockage est beaucoup plus faible que le nombre d'instances qui satisfont la charge de travail, ce qui explique pourquoi le stockage est peu utilisé alors que les unités de traitement sont fortement utilisées. Il est facile de voir que pour la plupart des métriques étudiées, les résultats par politique d'ordonnancement se chevauchent et pour cause, les requêtes dans ce cas sont homogènes.

TABLE 1 – Résultats de Adirbs, donnant des informations sur la puissance de calcul (PC), le nombre minimum d'instances requis (NoMIR), le nombre maximum d'instances requis (NoMaIR), le nombre effectif d'instances requis (NEIR) ainsi que leurs makespans correspondant respectifs (MNoMIR, MNoMaIR et MNEIR) et leurs coût correspondant respectifs (CNoMIR, CNoMaIR et CNEIR). Ce tableau donne aussi les Seuils d'utilisation du stockage et de la puissance de calcul (SUS et SUPC).

Stockage (Go)	**PC** (Gf)	**Algorithme**	**VM**	**NoMIR**
17.08	258.4	Fifo	d2-2	2

CNoMIR (euro)	**MNoMIR** (s)	**NoMaIR**	**CNoMaIR** (euro)	**MNoMaIR (s)**
439	86648576	4096	61298	5999534.1

NEIR	**CNEIR (euro)**	**MNEIR (s)**	**SUS (%)**	**SUPC (%)**
33	641.32	8016554.04	2.14	97.73

6. Conclusion

Dans cet article, nous avons présenté une approche (Adirbs) qui décrit comment concevoir et dimensionner une PFR, implémenté, à travers un cas d'utilisation (AiBle), mettant en évidence l'émission de requêtes homogènes sur le Cloud. A la question de savoir comment dimensionner efficacement l'infrastructure de ce cas d'utilisation, Adirbs apporte une réponse IaaS. Adirbs se base sur l'identification, la modélisation, la simulation des workflows, pour donner une réponse efficace, dans le but de réduire le plus efficacement possible la durée de l'activité et le coût d'utilisation des ressources sur la PFR via un principe se basant sur le front de Pareto. Grâce à son algorithme, Adirbs réduit considérablement le makespan global de la PFR, l'écart-type des charges de travail et les capacités de stockage, offrant ainsi un excellent compromis pour toutes ces métriques. Le modèle Adirbs renvoie ensuite le nombre minimal de ressources, leurs configurations matérielles et la politique d'ordonnancement des requêtes sur celles-ci, afin de réduire le nombre de ressources, le makespan et d'équilibrer au mieux les charges de travail et les stockages de ressources. Cependant, nous n'avons éprouvé Adirbs qu'avec des ressources et des requêtes homogènes . Dans les travaux futurs, nous étudierons également le cas où les requêtes et les ressources sont hétérogènes et mettrons en œuvre un équivalent réel de Adirbs.

Bibliographie

1. Ahmed (A.) et Sabyasachi (A. S.). – Cloud computing simulators : A detailed survey and future direction. – In *2014 IEEE international advance computing conference (IACC)*, pp. 866–872. IEEE, 2014.
2. Bhoi (U.), Ramanuj (P. N.) et al. – Enhanced max-min task scheduling algorithm in cloud computing. *International Journal of Application or Innovation in Engineering and Management (IJAIEM)*, vol. 2, n4, 2013, pp. 259–264.
3. Casanova (H.), Giersch (A.), Legrand (A.), Quinson (M.) et Suter (F.). – Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, vol. 74, n10, juin 2014, pp. 2899–2917.
4. Christou (P.), Simillidou (A.) et Stylianou (M. C.). – Tourists' perceptions regarding the use of anthropomorphic robots in tourism and hospitality. *International Journal of Contemporary Hospitality Management*, 2020.
5. Elzeki (O.), Rashad (M.) et Elsoud (M.). – Overview of scheduling tasks in distributed computing systems. *International Journal of Soft Computing and Engineering*, vol. 2, n3, 2012, pp. 470–475.
6. Etminani (K.) et Naghibzadeh (M.). – A min-min max-min selective algorihtm for grid task scheduling. – In *2007 3rd IEEE/IFIP International Conference in Central Asia on Internet*, pp. 1–7. IEEE, 2007.
7. Gudi (S. C.) et al. – Fog robotics : An introduction. – In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.
8. Johnson (S. G.). – The nlopt nonlinear-optimization package. – `http://github.com/stevengj/nlopt`.
9. Kapitonov (A.), Lonshakov (S.), Bulatov (V.), Kia (B.) et White (J.). – Robot-as-a-service : From cloud to peering technologies. – In *2021 The 4th International Conference on Information Science and Systems*, pp. 126–131, 2021.
10. Koubaa (A.). – A service-oriented architecture for virtualizing robots in robot-as-a-service clouds. – In *International Conference on Architecture of Computing Systems*, pp. 196–208. Springer, 2014.
11. Koubâa (A.) et al. – *Robot Operating System (ROS).* – Springer, 2017, 196–208p.
12. Kwon (M.), Biyik (E.), Talati (A.), Bhasin (K.), Losey (D. P.) et Sadigh (D.). – When humans aren't optimal : Robots that collaborate with risk-aware humans. – In *2020 15th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 43–52. IEEE, 2020.
13. Li (J.), Ma (T.), Tang (M.), Shen (W.) et Jin (Y.). – Improved fifo scheduling algorithm based on fuzzy clustering in cloud computing. *Information*, vol. 8, n1, 2017, p. 25.
14. Mazhar (B.), Jalil (R.), Khalid (J.), Amir (M.), Ali (S.) et Malik (B. H.). – Comparison of task scheduling algorithms in cloud environment. *International Journal of Advanced Computer Science and Applications*, vol. 9, n5, 2018, pp. 384–390.
15. Moniz (A. B.) et Krings (B.-J.). – Robots working with humans or humans working with robots ? searching for social dimensions in new human-robot interaction in industry. *Societies*, vol. 6, n3, 2016, p. 23.
16. Mushunuri (V.), Kattepur (A.), Rath (H. K.) et Simha (A.). – Resource optimization in fog enabled iot deployments. – In *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, pp. 6–13. IEEE, 2017.
17. Nezami (Z.), Zamanifar (K.), Djemame (K.) et Pournaras (E.). – Decentralized edge-to-cloud load balancing : Service placement for the internet of things. *IEEE Access*, vol. 9, 2021, pp. 64983–65000.

18. Norman (D. A.) et Bobrow (D. G.). – On data-limited and resource-limited processes. *Cognitive psychology*, vol. 7, n1, 1975, pp. 44–64.
19. Potluri (S.) et Rao (K. S.). – Quality of service based task scheduling algorithms in cloud computing. *International Journal of Electrical and Computer Engineering*, vol. 7, n2, 2017, p. 1088.
20. Quigley (M.), Conley (K.), Gerkey (B.), Faust (J.), Foote (T.), Leibs (J.), Wheeler (R.), Ng (A. Y.) et al. – Ros : an open-source robot operating system. – In *ICRA workshop on open source software*, number 3.2, p. 5. Kobe, Japan, 2009.
21. Ray (P. P.). – Internet of robotic things : Concept, technologies, and challenges. *IEEE access*, vol. 4, 2016, pp. 9489–9500.
22. Salot (P.). – A survey of various scheduling algorithm in cloud computing environment. *International Journal of Research in Engineering and Technology*, vol. 2, n2, 2013, pp. 131–135.