Compas'2018 : Parallélisme / Architecture / Système
ENSEEIHT - Toulouse, France, du 03 au 06 juillet 2018

Ordonnancement de tâches indépendantes sur machine parallèle sous contraintes d'énergie renouvelable

Ayham Kassab, Jean-Marc Nicod, Laurent Philippe, Veronika Rehn-Sonigo

Institut FEMTO-ST, Université Bourgogne Franche-Comté/CNRS/ENSMM – 25000 Besançon
[ayham.kassab|jean-marc.nicod|laurent.philippe|veronika.sonigo]@femto-st.fr

---

Résumé
Le développement continu des services et outils informatiques engendre une consommation électrique croissante qui participe au réchauffement climatique. Par ailleurs la concentration des moyens de stockage et de calcul au sein de centres de données ou centres de calcul fait que la consommation électrique de ces derniers prend une part toujours croissante dans la consommation globale. C'est pourquoi de nombreux travaux visent à la réduction de la consommation énergétique. Dans cet article nous nous intéressons au problème de l'alimentation d'un centre de calcul par des sources d'énergie renouvelables. Le problème consiste donc à utiliser au mieux l'énergie produite pour minimiser les temps de traitements, un challenge vu le comportement intermittent de la production. Nous proposons une évaluation de plusieurs algorithmes pour ordonnancer un ensemble de tâches indépendantes dont les caractéristiques sont connues à l'avance, sous contraintes de puissance. La contribution décrite est la mise en place de procédures de génération de données, de tâches et d'intervalles de puissance, pour comparer les performances de ces algorithmes par rapport à l'optimal et une étude par simulation des performances de plusieurs algorithmes.

Mots-clés : énergie renouvelable, ordonnancement, optimisation, Green computing, makespan

---

1. Introduction

Réduire l'emprunte écologique des activités humaines est un enjeu majeur dans la lutte contre le réchauffement climatique. Contrairement aux idées reçues, l'informatique est une industrie qui consomme de plus en plus d'énergie. En effet, les centres de données et les réseaux ont englouti en 2015 4% de la consommation énergétique mondiale. Or une très grande partie de cette énergie provient de centrales électriques fonctionnant grâce à des énergies fossiles, générant donc une empreinte carbone non négligeable.
Pour remédier à la situation, de nombreux efforts visent à réduire la consommation énergétique des composants de calcul [12], du système de refroidissement [17], ou du transport de l'énergie [2]. Dans ces domaines [14–16] proposent un tour d'horizon des outils et technologies utiles à la diminution de la consommation énergétique des centres de calcul. De même, une proposition souvent faite pour réduire la consommation d'un processeur est de ralentir ses performances de calcul par la réduction de sa fréquence et de son voltage (DVFS pour *Dynamic Voltage and Frequency Scaling*) [19]. Il est alors possible de profiter des temps morts pour que cette réduction n'ait pas d'impact négatif pour les calculs [18]. Les gains obtenus dans ce cas peuvent aller jusqu'à 70% pour un ralentissement de l'application de seulement 30%. Une

variante de DVFS, DVS pour *Dynamic Voltage Scaling*, est utilisée par Garg et al. [6] pour des applications haute performance dans des centres de données de type *Cloud* avec une diminution de la consommation énergétique pouvant aller jusqu'à 25%. Aujourd'hui, DVFS est intégré directement dans la conception des nouveaux processeurs [10,12].

Diminuer la consommation des systèmes informatiques n'est cependant qu'une partie de la réponse à apporter puisque leur utilisation continue de croître. L'enjeu de la réduction de l'emprunte écologique des systèmes informatiques ne s'arrête pas en effet à augmenter l'efficacité des centres de calcul mais s'accompagne également du choix de la provenance de l'énergie consommée. L'idéal serait qu'elle provienne uniquement de sources renouvelables. Cette option est très réaliste puisque le kWh est moins cher s'il est issu d'une centrale solaire que s'il est issu d'une centrale à charbon [4] et d'ici 2040 les coûts de production énergétique à partir du soleil ou du vent vont être plus que divisées par deux.

Le principal frein à l'utilisation de ce type d'énergie est le fait que la production est par nature intermittente. Dans [1], les auteurs montrent l'importance de la prédiction de la production issue des énergies renouvelables. Un ordonnancement des tâches peut alors être proposé afin de réduire le nombre de tâches échouées tout en améliorant le pourcentage de l'utilisation des énergies renouvelables. Parallèlement Goiri et al. [7,8] proposent un ordonnanceur nommé GreenSlot qui prend en compte les prédictions d'énergie renouvelable afin de lancer le plus de tâches quand l'alimentation en énergie verte est élevée. Mais le système a toujours recours au réseau électrique classique. Le projet de recherche ANR DATAZERO [5] a pour objectif de proposer une architecture matérielle complète d'un centre de données uniquement alimenté par des énergies renouvelables [9]. Il propose une boucle de négociation pour faire coïncider les objectifs antinomique de la partie calcul [3] et la partie électrique [11] qui préfère stocker pour atténuer la variabilité de la production électrique et de la demande de ressources informatiques.

Dans cet article, nous nous intéressons à la manière d'ordonnancer des tâches séquentielles indépendantes sur une architecture parallèle dont l'alimentation contrainte est variable au cours du temps, ce qui est le cas lorsque l'électricité est produite à partir d'énergies renouvelables comme le vent et le soleil. Ainsi, les tâches doivent être placées à des moments où la puissance électrique est suffisante pour permettre leur exécution. À ce titre notre travail se différencie de l'approche classique sur la maîtrise de l'énergie car la contrainte à satisfaire est que les tâches en cours ne consomment pas plus que la puissance instantanée disponible alors que plus classiquement la contrainte est l'énergie donc l'intégrale de la puissance sur le temps. Nous visons plutôt les centres de calcul qui peuvent tolérer la variation, car les tâches peuvent être différées, à la différence des centres de données qui fournissent des services en ligne à exécuter dans l'instant. La contribution se situe dans la proposition d'algorithmes adaptés, dans la mise en place de procédures de génération de données, tâches et intervalles de puissance, pour comparer les performances de ces algorithmes par rapport à l'optimal et, dans une étude par simulation des performances des algorithmes.

La suite de cet article est organisée de la manière suivante. Le paragraphe 2 décrit le modèle utilisé ainsi que le problème à résoudre. Le paragraphe 3 décrit les algorithmes proposés dans ce nouveau contexte. La description des méthodes de génération des ensembles de tâches et intervalles pour rendre possible les comparaisons à un optimal connu et des enveloppes énergétiques disponibles fait l'objet du paragraphe 4. La description des simulations et les discussions des résultats est faite dans le paragraphe 5. L'article se termine par une conclusion et des perspectives.

2. Modèle

Dans ce travail nous nous intéressons au problème d'ordonnancer un ensemble $\mathcal{T} = \{T_1, \ldots, T_n\}$ de n tâches séquentielles indépendantes sur un ensemble $\mathcal{M} = \{M_1, \ldots, M_m\}$ de m machines sous des contraintes de puissance électrique. Une tâche T_i, $1 \leq i \leq n$, a un temps de calcul p_i et un besoin de puissance de électrique φ_i pour être exécutée. Les tâches ne sont pas préemptibles. L'ensemble des tâches T_i doit être ordonnancé sur un ensemble d'unités de calcul $\mathcal{M}$. L'énergie consommée par une machine comporte une partie statique et une partie dynamique ainsi que des phases de démarrage et d'extinction. Pour simplifier le modèle, nous considérons dans ce travail que les unités de calcul M_j sont des coeurs appartenant à une machine parallèle à mémoire partagée. Enfin, pour cette étude, nous faisons l'hypothèse que toutes les unités de calcul sont exclusivement alimentées par des sources d'énergie renouvelables telles que des éoliennes et des panneaux solaires. Le comportement intermittent de ces sources fait que la production énergétique varie au cours du temps et nous la modélisons via un ensemble $\Delta = \{\Delta_1, \Delta_2, \ldots, \Delta_X\}$ de X intervalles Δ_x de longueur δ_x au cours duquel la puissance électrique disponible est Φ_x. Notre objectif est de trouver un ordonnancement de l'ensemble des tâches $\mathcal{T}$ sur les unités de calcul M_j qui termine le plus tôt possible tout en respectant les contraintes énergétiques, c'est-à-dire nous visons à optimiser le *makespan*. Dans la classification de Graham, complétée en [13], le problème est décrit par $P|\varphi_i \leq \Phi_x|C_{max}$.

Un exemple d'ordonnancement possible est montré dans la figure 1. Ici, les tâches sont ordonnancées dans l'ordre de leur numéro. On remarque que la tâche T_1 ne peut pas commencer en premier dans l'intervalle Δ_1 parce que sa demande en puissance énergétique φ_1 dépasse la puissance énergétique disponible Φ_1 de l'intervalle Δ_1. Elle est donc ordonnancée plus tard dans l'intervalle Δ_2 qui respecte la contrainte énergétique de T_1.

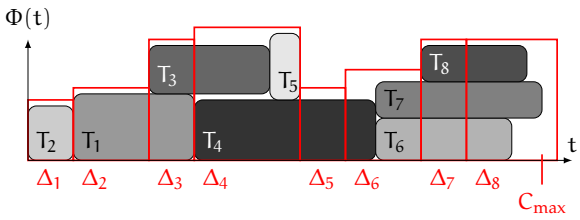


FIGURE 1 – Exemple d'ordonnancement

3. Algorithmes

Du fait du format court des articles nous avons choisi ici de nous limiter aux algorithmes les plus simples. Nous verrons par la suite que ces derniers obtiennent néanmoins de bonnes performances, ce qui justifie notre choix.

Les algorithmes considérés sont des algorithmes reposant tous sur un algorithme de liste. Un algorithme de liste est un algorithme qui prend les tâches une par une dans une liste et les positionne au plus tôt dans l'ordonnancement, c'est un algorithme glouton. La différence entre deux algorithmes vient uniquement de la manière de trier la liste suivant les priorités associées aux tâches. Dans notre problème deux données caractérisent les tâches et peuvent donc être utilisées comme priorité : le temps d'exécution p_i et le besoin de puissance φ_i. Les tâches les plus difficiles à ordonnancer, celles qui devront donc l'être en premier, sont celles qui ont des valeurs de p_i ou de φ_i importantes. Nous proposons trois priorités possibles : p_i, qui donne l'algorithme classique LPT (Largest Processing Time), φ_i, qui donne l'algorithme LPN (Largest Power Need) et, $p_i \times \varphi_i$, qui donne l'algorithme LPTPN tenant compte à la fois des deux valeurs. Les premières expériences nous ont montrés que ces critères permettent de gérer les tâches longues (LPT), les tâches ayant un besoin de puissance important (LPN) ou les tâches longues avec un besoin de puissance important (LPTPN). Ils ne permettent pas de gérer le cas

plus général où nous avons à la fois des tâches longues, mais qui n'ont pas un gros besoin en puissance énergétique, et des tâches ayant un besoin important de puissance mais qui durent peu de temps. Nous proposons donc un quatrième algorithme (2Qs) qui repose sur deux files, l'une triée par p_i et l'autre par $p_i \times \varphi_i$ décroissants, les priorités qui donnaient les meilleurs résultats dans nos premières expériences, et prend les tâches alternativement dans les deux files.

4. Génération de tâches et d'intervalles

L'évaluation d'heuristiques est complexe pour un problème NP-Complet car nous n'avons pas de valeur optimale à laquelle comparer les résultats obtenus. Il est parfois possible de trouver des bornes inférieures. Dans notre problème, une borne du makespan est la valeur définissant la surface $\sum_{x=0}^{C_{max}} \Phi_x = \sum_{i=0}^{N}(p_i \times \varphi_i)$, c'est-à-dire une surface d'énergie disponible équivalente à la somme des besoins des tâches. Mais cette borne ne prend pas en compte la répartition de la puissance entre les intervalles alors qu'il est plus facile de réaliser un ordonnancement si la puissance disponible est constante que si elle est très variable. Pour comparer les performances des algorithmes par rapport à l'optimal nous proposons deux méthodes : la génération de tâches à partir d'un ensemble d'intervalles (TFI, Tasks From Intervals) et la génération d'un ensemble d'intervalles à partir de tâches (IFT, Intervals From Tasks).
Pour la méthode TFI nous tirons aléatoirement un point de départ de la tâche parmi les intervalles, puis une puissance à partir de la puissance disponible à l'instant choisi et nous calculons la durée pendant laquelle cette puissance est maintenue. Sur la surface ainsi définie nous créons aléatoirement une tâche, qui est ajoutée à la liste des tâches, puis l'énergie correspondant à cette tâche est supprimée des intervalles utilisés. L'algorithme fonctionne jusqu'à ce que toute la puissance disponible soit utilisée. À noter que, pour remplir les trous, cet algorithme tend à créer des petites tâches. Des arrondis sont donc réalisés pour éviter d'en générer trop.
La méthode IFT est plus simple puisqu'il suffit de positionner aléatoirement les tâches et d'ajouter les intervalles correspondants. À la première création de tâche, un intervalle est créé qui correspond exactement aux besoins de la tâche. Lors de la création de la tâche suivante, soit il n'y a pas de recouvrement entre les deux tâches et un seul nouvel intervalle est créé, soit de nouveaux intervalles sont créés pour les durées où la première tâche est seule, les deux tâches s'exécutent en même temps, et la seconde tâche s'exécute seule. Les tâches suivantes sont ajoutées sur le même principe, en vérifiant que le nombre de cœurs de la machine n'est pas dépassé. Cet algorithme génère des ensembles d'intervalles de longueurs δ_x irrégulières.
À noter que les deux algorithmes génèrent des cas particuliers puisque ce sont des cas avec une solution où les tâches peuvent consommer entièrement la puissance d'entrée. Il n'est donc pas exclu que ces cas introduisent un biais dans les résultats.

5. Résultats

Nous donnons dans cette partie les conditions et les résultats de simulation. Pour évaluer la performance des algorithmes que nous avons développés en python, nous considérons des jeux de tâches et des jeux d'intervalles différents. L'objectif est de minimiser le temps de fin de calcul de la dernière tâche (makespan). Comme cette valeur dépend de la taille des tâches et de la puissance disponible dans les intervalles, nous normalisons les résultats en calculant la mesure $PerMake = (makespan - inutilisable)/\sum(p_i)$ où $inutilisable$ est la somme des longueurs des intervalles avec $\Phi_x < min_i(\varphi_i)$. La performance des algorithmes est évaluée sur des valeurs différentes de p_i et φ_i maximales. Toutes les simulations sont effectuées dans le cas d'une machine à 8 cœurs. Nous avons développé trois expériences avec des paramètres

différents.

La première expérience sert à déterminer quel algorithme obtient la valeur de $perMake$ minimale. Les valeurs de p_i sont générées aléatoirement suivant une loi exponentielle de telle sorte que la valeur de p_{max} va de 10 à 100 par pas de 10. Les valeurs de φ_i sont générées aléatoirement suivant une loi uniforme avec la valeur de φ_{max} qui va de 4 à 40 par pas de 4. La valeur maximale de la puissance disponible dans les intervalles est alors $\Phi_{max} = 80$. La figure 2 montre le meilleur algorithme après avoir effectué 200 simulations pour chaque couple (p_{max}, Φmax), sur des ensembles différents de 100 tâches et des ensembles de 1000 intervalles de longueur $\delta_x = 10$. Ces résultats ont été donnés dans une autre publication [13] mais la figure présentée est une version améliorée qui inclut l'algorithme 2Qs et qui se base sur 200 itérations au lieu de 100. Nous pouvons dire que lorsque Φ_{max} est inférieur à 28, 2Qs est le meilleur. Quand Φ_{max} augmente, il devient moins intéressant d'utiliser l'ordre de la file de LPT : LPTPN devient le meilleur.

Les deux expériences qui suivent servent à comparer les performances des algorithmes par rapport à l'optimal, avec les méthodes de génération IFT et TFI. La distance par rapport à l'optimal est calculée comme suit, $OPT_{dis} = C_{max} / C^{opt}_{max}$, où C^{opt}_{max} est le makespan obtenu par la méthode de génération.

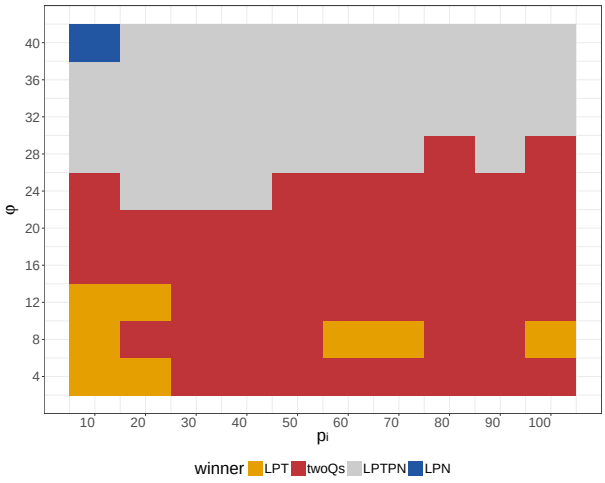
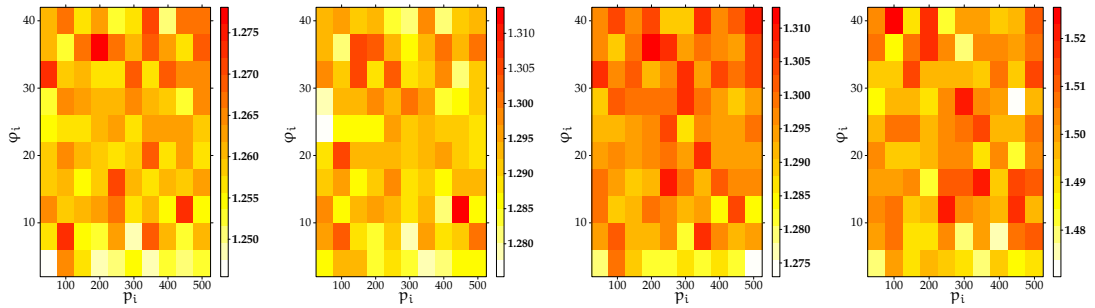


FIGURE 2 – La meilleure valeur de $perMake$

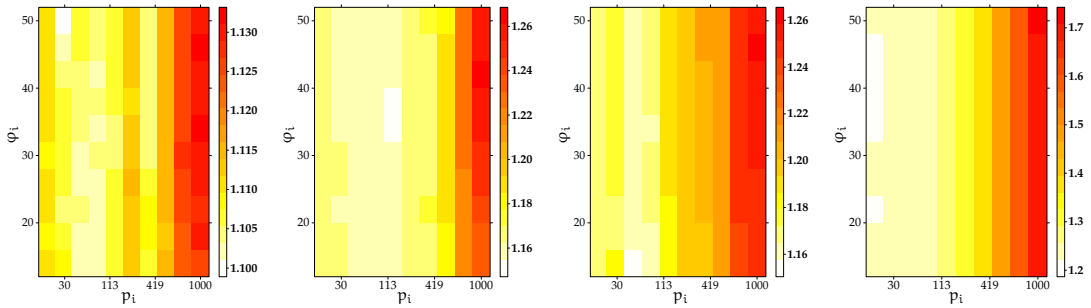
Pour la méthode IFT, des tâches avec des p_i plus longs sont nécessaires afin d'avoir une enveloppe d'énergie appropriée produite à partir de l'ensemble des tâches : p_{max} va de 50 à 500 par pas de 50. La génération de φ_i est la même que dans la première expérience. Pour les intervalles nous prenons l'ensemble généré par la méthode IFT dont la taille est C^{opt}_{max}, que nous reproduisons trois fois pour nous assurer qu'il y a assez de place pour exécuter toutes les tâches. La figure 3 montre la distance à l'optimal pour chaque algorithme. L'échelle utilisée sur chacune des "heatmaps" est différente, ce qui engendre des résultats qui semblent similaires mais ne le sont pas en réalité. Nous pouvons ainsi constater que 2Qs produit les solutions les plus proches de l'optimal, entre 1.247 et 1.276 et que LPN produit de moins bonnes solutions, entre 1.474 et 1.523. À noter que, quelque soient les valeurs de p_i et φ_i, la distance à l'optimal varie peu, entre 2 et 4 %, mais qu'elle varie plus pour LPN que pour 2Qs.



(a) 2Qs (b) LPTPN (c) LPT (d) LPN

FIGURE 3 – Distance à l'optimal avec la méthode IFT

Pour la méthode TFI, nous avons exploré des valeurs de p_i jusqu'à 1000, p_i prend des valeurs logarithmiques ($p_{max} \in \{20, 30, 47, 73, 113, 175, 271, 419, 647, 1000\}$) pour améliorer la visibilité. La valeur φ_i est calculée par une loi uniforme avec la valeur de φ_{max} qui va de 14 à 50 par pas de 4. L'optimal C_{max}^{opt} est la fin de l'ensemble des $nbInter$ intervalles utilisés pour générer les tâches, $nbInter \in \{9, 10, 13, 15, 17, 19, 21, 23, 25, 27\}$ et la longueur est fixée à 60. La distance à l'optimal obtenue est donnée à la figure 4. Comme pour la figure précédente, nous n'avons pas unifié les échelles pour mieux mettre en valeur les propriétés de chaque algorithme. Les résultats de la figure 4 sont confirmés ici : les solutions produites par 2Qs sont les plus proches de l'optimal et LPN est l'algorithme le moins performant. Avec la méthode TFI, la distance à l'optimale est plus faible qu'avec la méthode IFT, et l'algorithme 2Qs est au maximum à 13% de la valeur optimale. Nous pouvons également déduire du dégradé des heatmaps que le problème devient plus difficile lorsque la taille des tâches augmente. Cette perte d'efficacité est faible, limitée à 3%, dans le cas de l'algorithme 2Qs mais plus importante, jusqu'à 50% pour LPN.



(a) 2Qs (b) LPTPN (c) LPT (d) LPN

FIGURE 4 – Distance à l'optimal avec la méthode TFI

Tous les algorithmes à base de liste simple s'exécutent en environ 0.02 sec. L'algorithme 2Qs prend deux fois plus de temps. Les temps restent très raisonnables même pour envisager un ordonnancement en ligne.

6. Conclusion

Nous avons présenté l'étude des performances de plusieurs algorithmes de liste pour le problème de l'ordonnancement de tâches et intervalles quelconques. Nos résultats montrent que l'algorithme 2Qs donne les meilleurs résultats quand la valeur de Φ_{max} est inférieure à 28, sinon il faut privilégier l'algorithme LPTPN. Dans le cas où les jeux de données sont calculés à partir d'une solution optimale, 2Qs donne les solutions les plus proches de l'optimal, ce qui est confirmé par deux expériences différentes. Malgré son temps d'exécution deux fois plus lent que les autres algorithmes, il reste assez réactif pour l'utilisation en ordonnancement en ligne. Dans nos travaux futurs nous pensons, d'une part, utiliser la méthode de génération d'instances optimales pour comparer des algorithmes plus sophistiqués à la solution optimale et, d'autre part, étendre notre étude au cas des architectures distribuées.

Remerciements

Ce travail a été en partie supporté par le projet ANR DATAZERO (contrat "ANR-15-CE25-0012") et le projet Labex ACTION (contrat "ANR-11-LA BX-01-01"). Les calculs ont été effectués sur le calculateur du Mésocentre de calcul de Franche-Comté.

Bibliographie

1. Aksanli (B.), Venkatesh (J.), Zhang (L.) et Rosing (T.). – Utilizing green energy prediction to schedule mixed batch and service jobs in data centers. – In *Proceedings of the 4th Workshop on Power-Aware Computing and Systems*, *HotPower '11*, HotPower '11, pp. 5 :1–5 :5, New York, NY, USA, 2011. ACM.
2. Arlitt (M.), Bash (C.), Blagodurov (S.), Chen (Y.), Christian (T.), Gmach (D.), Hyser (C.), Kumari (N.), Liu (Z.), Marwah (M.) et al. – Towards the design and operation of net-zero energy data centers. – In *Thermal and Thermomechanical Phenomena in Electronic Systems (ITherm), 2012 13th IEEE Intersociety Conference on*, pp. 552–561. IEEE, 2012.
3. Caux (S.), Rostirolla (G.) et Stolf (P.). – Smart Datacenter Electrical Load Model for Renewable Sources Management. – In *International Conference on Renewable Energies and Power Quality (ICREPQ)*. European Association for the Development of Renewable Energies, Environment and Power Quality, 2018. – `http://www.icrepq.com/icrepq18/231-18-caux.pdf`.
4. Cheung (A.), Rooze (J.), Gandolfo (A.) et Marquina (D.). – *Beyond the tipping point*. – Rapport technique, Bloomberg New Energy Finance, November 2017. Available at https ://uk.eaton.com/content/content-beacon/RE-study/GB/home.html ?wtredirect=www.eaton.com/tippingpoints.
5. ANR DATAZERO. `http://www.datazero.org`.
6. Garg (S. K.), Yeo (C. S.), Anandasivam (A.) et Buyya (R.). – Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers. *Journal of Parallel and Distributed Computing*, vol. 71, n6, 2011, pp. 732 – 749.
7. Goiri (Í.), Haque (M. E.), Le (K.), Beauchea (R.), Nguyen (T. D.), Guitart (J.), Torres (J.) et Bianchini (R.). – Matching renewable energy supply and demand in green datacenters. *Ad Hoc Networks*, vol. 25, 2015, pp. 520–534.
8. Goiri (I.), Le (K.), Haque (M. E.), Beauchea (R.), Nguyen (T. D.), Guitart (J.), Torres (J.) et Bianchini (R.). – Greenslot : Scheduling energy consumption in green datacenters. *2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC 2011)*, vol. 00, 2012, pp. 1–11.
9. Grange (L.), Costa (G. D.) et Stolf (P.). – Green IT scheduling for data center powered with renewable energy. *Future Generation Computer Systems*, 2018.
10. Hackenberg (D.), Schöne (R.), Ilsche (T.), Molka (D.), Schuchart (J.) et Geyer (R.). – An energy efficiency feature survey of the intel haswell processor. – In *International Parallel and Distributed Processing Symposium Workshop (IPDPSW)*, pp. 896–904. IEEE, 2015.
11. Haddad (M.), Nicod (J.-M.) et Péra (M.-C.). – Data-center supply-refueling station synergy. – In *Vehicle Power and Propulsion Conference (VPPC 2017)*, pp. 1 – 6, Belfort, dec 2017. IEEE.
12. Hofmann (J.), Fey (D.), Eitzinger (J.), Hager (G.) et Wellein (G.). – Analysis of Intel's Haswell Microarchitecture Using The ECM Model and Microbenchmarks. – In *International Conference on Architecture of Computing Systems*, pp. 210–222. Springer, 2016.
13. Kassab (A.), Nicod (J. M.), Philippe (L.) et Rehn-Sonigo (V.). – Scheduling independent tasks in parallel under power constraints. – In *46th International Conference on Parallel Processing (ICPP)*, pp. 543–552, 2017.
14. Kaur (T.) et Chana (I.). – Energy efficiency techniques in cloud computing : A survey and taxonomy. *ACM Comput. Surv.*, vol. 48, n2, octobre 2015, pp. 22 :1–22 :46.
15. Orgerie (A.-C.), Assuncao (M. D. d.) et Lefevre (L.). – A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Comput. Surv.*, vol. 46, n4, mars 2014, pp. 47 :1–47 :31.
16. Oró (E.), Depoorter (V.), Garcia (A.) et Salom (J.). – Energy efficiency and renewable energy

integration in data centres. strategies and modelling review. *Renewable and Sustainable Energy Reviews*, vol. 42, 2015, pp. 429 – 445.

17. Patel (C.), Sharma (R.), Bash (C.) et Graupner (S.). – Energy aware grid : Global workload placement based on energy efficiency. – In *ASME 2003 International Mechanical Engineering Congress and Exposition*, pp. 267–275, 2003.
18. Wang (L.), Khan (S. U.), Chen (D.), KołOdziej (J.), Ranjan (R.), Xu (C.-Z.) et Zomaya (A.). – Energy-aware parallel task scheduling in a cluster. *Future Generation Computer Systems*, vol. 29, n7, 2013, pp. 1661–1670.
19. Wang (L.), Von Laszewski (G.), Dayal (J.) et Wang (F.). – Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS. – In *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pp. 368–377. IEEE Computer Society, 2010.