

Ordonnancement Des Workflows Scientifiques Sur Le Cloud Avec Optimisation De L'énergie

C.Houcine Abdelkader T.Abdelghani Chabane

Département d'Informatique
Université des sciences et de la technologie d'Oran - Mohamed Boudiaf

11 juin 2023

Sommaire

- ① Cloud Computing
- ② Workflow & Workflow Scientifique
- ③ Ordonnancement Des Workflow Scientifiques Sur Cloud
 - État de l'art
- ④ Conception De Notre Approche
- ⑤ Implémentation, Simulation & Discussion

Définition Du Cloud Computing

Définition

Cloud Computing est un modèle pour permettre un omniprésent, commode, accès à la demande à un parc partagé de ressources informatiques configurables (Réseaux, serveur, stockage, applications et services) qui peuvent être mis rapidement à disposition et libère avec une intervention et interaction minimale du fournisseur de services[6, 3].

Modèles De Déploiement

Il existe 5 modèles de déploiement de Cloud Computing[6][1] :

- Cloud privé.
- Cloud communautaire.
- Cloud publique.
- Cloud hybride.
- Multi-Cloud Computing.

Modèles De Déploiement

Il existe 5 modèles de déploiement de Cloud Computing[6][1] :

- Cloud privé.
- Cloud communautaire.
- Cloud publique.
- Cloud hybride.
- Multi-Cloud Computing.

Modèles De Déploiement

Il existe 5 modèles de déploiement de Cloud Computing[6][1] :

- Cloud privé.
- Cloud communautaire.
- Cloud publique.
- Cloud hybride.
- Multi-Cloud Computing.

Modèles De Déploiement

Il existe 5 modèles de déploiement de Cloud Computing[6][1] :

- Cloud privé.
- Cloud communautaire.
- Cloud publique.
- Cloud hybride.
- Multi-Cloud Computing.

Modèles De Déploiement

Il existe 5 modèles de déploiement de Cloud Computing[6][1] :

- Cloud privé.
- Cloud communautaire.
- Cloud publique.
- Cloud hybride.
- Multi-Cloud Computing.

Modèles De Services

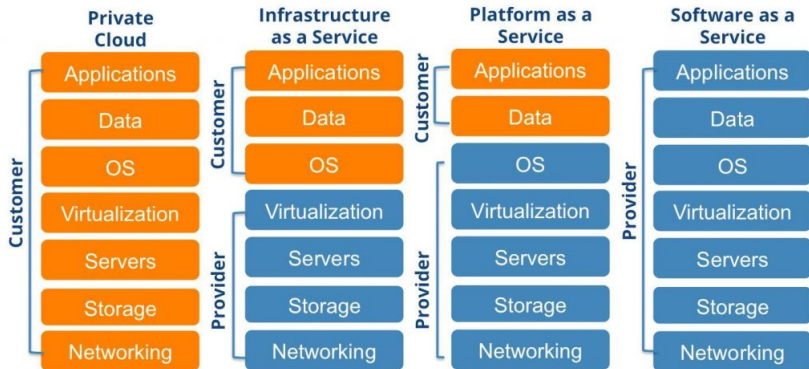


Figure – Les modèles des services Cloud quisted.net

Définition Du Workflow

Définition

Le workflow est la séquence de tâches, d'étapes et de décisions qui doivent être suivies pour mener à bien un processus spécifique. On peut le considérer comme un ensemble d'instructions qui décrit comment un processus doit être effectué, y compris l'ordre dans lequel les tâches doivent être effectuées, qui est responsable de l'exécution de chaque tâche et ce qui doit se passer ensuite en fonction du résultat de chaque tâche[7].

Définition Des Workflow Scientifiques

Définition

Les workflows scientifiques sont des applications gourmandes en données représentant des sources de données distribuées et des calculs complexes dans divers domaines, à savoir l'astronomie, la bio-informatique... Dans les environnements distribués, divers capteurs et processus expérimentaux génèrent un grand volume de données qui doivent être collectées et traitées dans des délais spécifiques[4].

Cycle De Vie Des Workflows Scientifiques

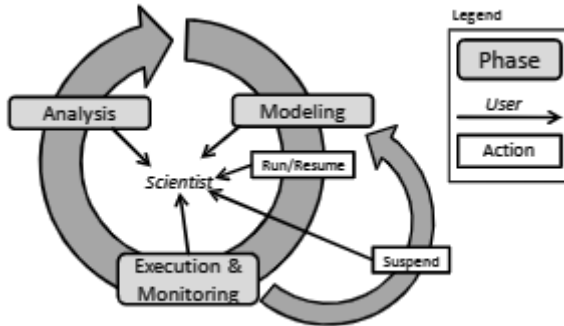


Figure – Cycle de vie des workflow scientifiques[2]

Ordonnancement Des Workflow Scientifiques

- La planification des tâches du workflow scientifique est extrêmement importante.
- Les algorithmes de planification sont fondés sur des politiques appropriées pour déployer le workflow scientifique aux machines virtuelles.
- Étant donné que les nœuds physiques sont hétérogènes et que leur état d'exécution change de façon dynamique au fil du temps, et que les ressources système requises par les utilisateurs et les types de demandes d'utilisateurs sont différents.

Ordonnancement Des Workflow Scientifiques

Une mauvaise planification peut entraîner :

- une baisse des performances du système
- une longue période de rétroaction des demandes
- une baisse de la satisfaction des utilisateurs

Donc l'objectif est :

- optimiser l'allocation des ressources
- minimiser le temps d'exécution
- minimiser les coûts d'exploitation
- maximiser la satisfaction des utilisateurs

Représentation De Workflow

- Les applications sont définies par des graphes acycliques dirigés (DAG). Un workflow comprend un ensemble de tâches interdépendantes qui sont liées entre elles par des données ou des dépendances fonctionnelles[5].
- Nous considérons le workflow W comme un graphique $G = (T, D)$, où $T = \{ T_0, T_1, \dots, T_n \}$ indique n tâches et indique les dépendances de flux de données entre elles. La dépendance (T_i, T_j) signifie que la tâche T_i est un prédécesseur immédiat de la tâche T_j et que la tâche T_j est un successeur immédiat de la tâche T_i .

Représentation De Workflow

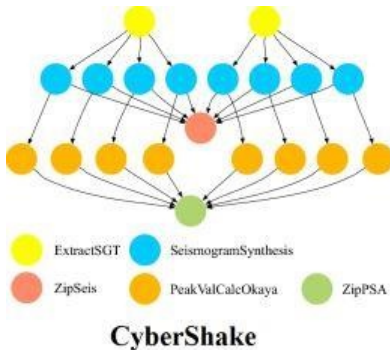


Figure – CyberShake [confluence.pegasus.isi.edu]

Allocation Des Machines Virtuelles

- *Types de VM*

$$\begin{cases} VM_{HEC} & (HEC_1), (HEC_2), (HEC_3).....HEC_k \\ VM_{MEC} & (MEC_1), (MEC_2), (MEC_3).....MEC_k \\ VM_{LEC} & (LEC_1), (LEC_2), (LEC_3).....LEC_k \end{cases}$$

- *Groupes des tâches*

$$\begin{cases} HCT & (HCT1), (HCT2), (HCT3).....HCT_n \\ MCT & (MCT1), (MCT2), (MCT3).....MCT_n \\ LCT & (LCT1), (LCT2), (LCT3).....LCT_n \end{cases}$$

Lignes directrices

- 1 Cloud Computing
- 2 Workflow & Workflow Scientifique
- 3 Ordonnancement Des Workflow Scientifiques Sur Cloud
 - État de l'art
- 4 Conception De Notre Approche
- 5 Implémentation, Simulation & Discussion

Objectives

- Notre algorithme considère les métriques suivantes :
 - Consommation de l'énergie
 - Coût
 - Makespan
- Notre algorithme considère les contraintes suivantes :
 - Les dépendances entre les tâches
 - Délai

Objectives

- Notre algorithme considère les métriques suivantes :
 - Consommation de l'énergie
 - Coût
 - Makespan
- Notre algorithme considère les contraintes suivantes :
 - Les dépendances entre les tâches
 - Délai

Description

Notre approche consiste à proposer un nouvel algorithme MOCS-OViC (Multi-Objective Cloud Scheduler with Optimized Virtual Machines Consolidation for Scientific Workflows) qui considère deux étapes principales pour la planification :

- 1 affecter des tâches aux machines virtuelles (VM) appropriées. Pour cette étape, nous appliquons l'algorithme EViMA.
- 2 appliquer certaines stratégies de migration de VM au cours de l'étape d'exécution, pour éviter le problème des machines physiques (PMs) sous-chargées ou surchargées qui consomment plus d'énergie.

Description

Notre approche consiste à proposer un nouvel algorithme MOCS-OViC (Multi-Objective Cloud Scheduler with Optimized Virtual Machines Consolidation for Scientific Workflows) qui considère deux étapes principales pour la planification :

- 1 affecter des tâches aux machines virtuelles (VM) appropriées. Pour cette étape, nous appliquons l'algorithme EViMA.
- 2 appliquer certaines stratégies de migration de VM au cours de l'étape d'exécution, pour éviter le problème des machines physiques (PMs) sous-chargées ou surchargées qui consomment plus d'énergie.

Algorithmme EViMA

Algorithm 4.1 EViMA

Require: Workflow, set of VMs and set of VM types ($VM_{HEC}, VM_{MEC}, VM_{LEC}$)

```
1:  $wt_{ReadyPool} = clustered(wt_1, wt_2, wt_3, \dots, wt_n)$ 
2: while  $wt_{ReadyPool} \neq \phi$  do
3:   Compute EFT of each tasks
4:   Group  $Tasks \Rightarrow HCT, MCT, LCT$ 
5:   Group  $VMs \Rightarrow (VM_{HEC}, VM_{MEC}, VM_{LEC})$ 
6:   foreach  $wt$  in  $wt_{ReadyPool}$  do
7:     if  $wt_i$  in HCT and  $wt_j$  in LCT then
8:       apply algorithm 2 to execute HCT
9:       apply algorithm 3 to execute LCT
10:    else
11:      if EFT of  $wt_i = DI$  of  $wt_i$  then
12:         $wt_i \mapsto VM_{MEC}$ 
13:        Update  $wt_{ReadyPool}$ 
14:      else
15:        if EFT of  $wt_i \leq ST$  then
16:          apply algorithm 5
17:        else
18:          apply algorithm 4 to save mood the idle VM
19:          Update  $wt_{ReadyPool}$ 
20:        end if
21:      end if
22:    end if
23:  end foreach
24: end while
```

L'emplacement Et La Consolidation Des Machines Virtuelles

Il y a 4 sous problèmes :

- Détection des hôtes sous-chargés : [boxplot](#)
- Détection des hôtes surchargés : [boxplot](#)
- Sélectionner une machine virtuelle : [Minimum Energy Cost Migration \(MECM\)](#).
- Sélectionner un hôte de destination : [algorithme glouton](#)

L'emplacement Et La Consolidation Des Machines Virtuelles

Il y a 4 sous problèmes :

- Détection des hôtes sous-chargés : [boxplot](#)
- Détection des hôtes surchargés : [boxplot](#)
- Sélectionner une machine virtuelle : [Minimum Energy Cost Migration \(MECM\)](#).
- Sélectionner un hôte de destination : [algorithme glouton](#)

L'emplacement Et La Consolidation Des Machines Virtuelles

Il y a 4 sous problèmes :

- Détection des hôtes sous-chargés : [boxplot](#)
- Détection des hôtes surchargés : [boxplot](#)
- Sélectionner une machine virtuelle : [Minimum Energy Cost Migration \(MECM\)](#).
- Sélectionner un hôte de destination : [algorithme glouton](#)

L'emplacement Et La Consolidation Des Machines Virtuelles

Il y a 4 sous problèmes :

- Détection des hôtes sous-chargés : [boxplot](#)
- Détection des hôtes surchargés : [boxplot](#)
- Sélectionner une machine virtuelle : [Minimum Energy Cost Migration \(MECM\)](#).
- Sélectionner un hôte de destination : [algorithme glouton](#)

MOCS-OViC

Algorithm 4.6 MOCS-OViC

Require: Workflow, set of VMs and set of VM types ($VM_{HEC}, VM_{MEC}, VM_{LEC}$), PMs

- 1: $wt_{ReadyPool} = clustered(wt_1, wt_2, wt_3, \dots, wt_n)$
 - 2: Apply algorithm EViMA
 - 3: Sort PMs based on capacity /availability in descending order
 - 4: $VM_{candidateList} = \phi$
 - 5: Add all VMs that are in under-loaded PMs to $VM_{candidateList}$
 - 6: Add all VMs that are in over-loaded PMs to $VM_{candidateList}$
 - 7: $j=0$
 - 8: **while** $VM_{candidateList} \neq \phi$ and $j < \text{size}(PMs)$ **do**
 - 9: pick PM_j
 - 10: $VM_{selected}$ which results minimum energy is selected
 - 11: **if** $\text{requirements}(VM_{selected}) \leq \text{availableCapacity}(PM_j)$ **then**
 - 12: place $VM_{selected}$ on PM_j
 - 13: Update $VM_{candidateList}$
 - 14: **end if**
 - 15: $j=j+1$
 - 16: **end while** Until no more VM or no more available capacities on PMs
 - 17: Migrate all VMs
 - 18: **foreach** pm in PMs **do**
 - 19: **if** pm is not used **then**
 - 20: put pm on power save mode
 - 21: **end if**
 - 22: **end foreach**
-

Applications Workflow

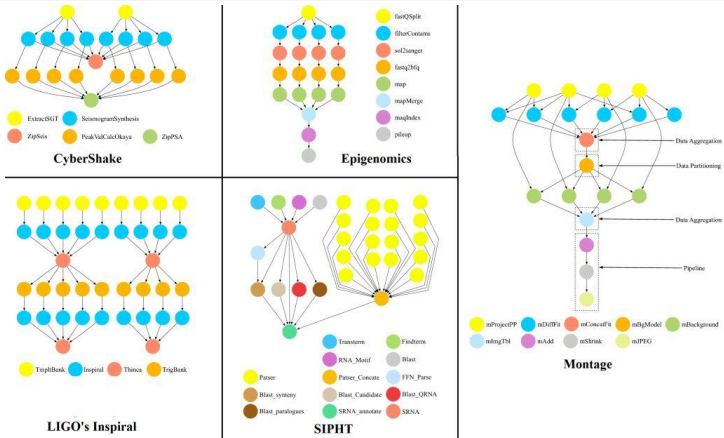


Figure – Les Workflows utilisés [confluence.pegasus.isi.edu]

Les Workflows De Petite Taille

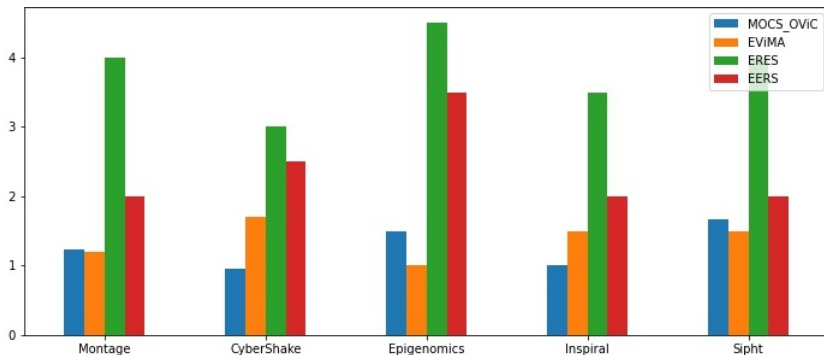


Figure – La consommation d'énergie dans les workflow de petite taille

Les Workflows De Taille Moyenne

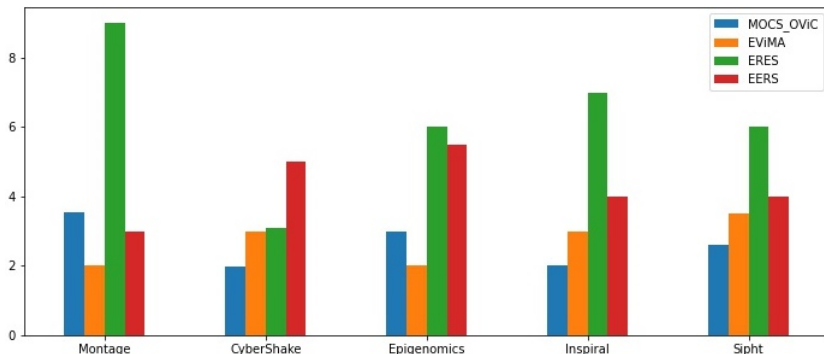


Figure – La consommation d'énergie dans les workflow de taille moyenne

Les Workflows De Grande Taille

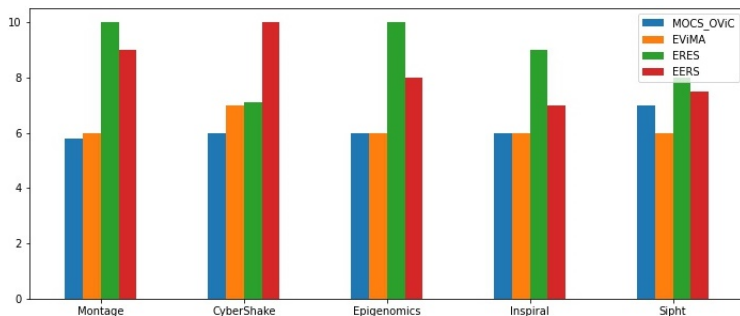


Figure – La consommation d'énergie dans les workflow de grande taille

Les Workflows De Très Grande Taille

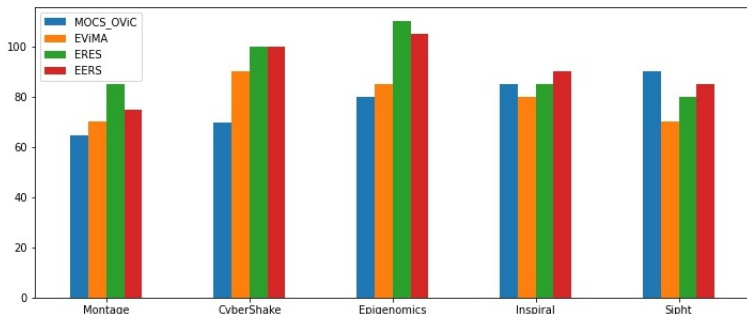


Figure – La consommation dans les workflows de très grande taille

Comparaison

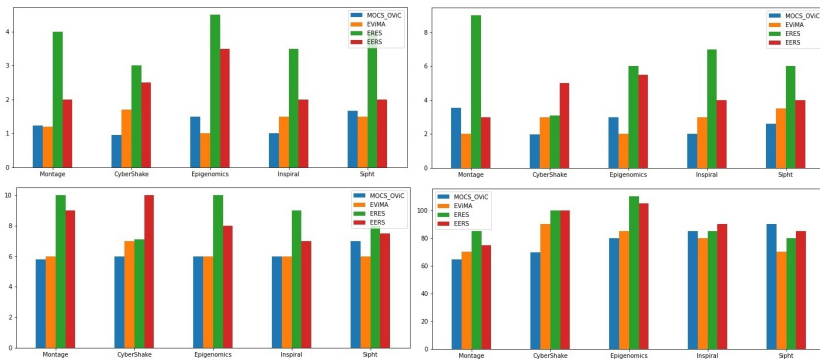


Figure – La comparaison de la consommation de l'énergie

Discussion

Avantage

Notre approche MOCS-OViC consomme moins d'énergie que les algorithmes ERES et EERS à tout moment sauf en 2 cas.

Inconvénient

MOCS-OViC peut économiser plus d'énergie que EViMA lorsque CyberShake et Inspiral sont exécutés pour toutes tailles et pour Epigenomics de très grande taille.

Les Workflows De Petite Taille

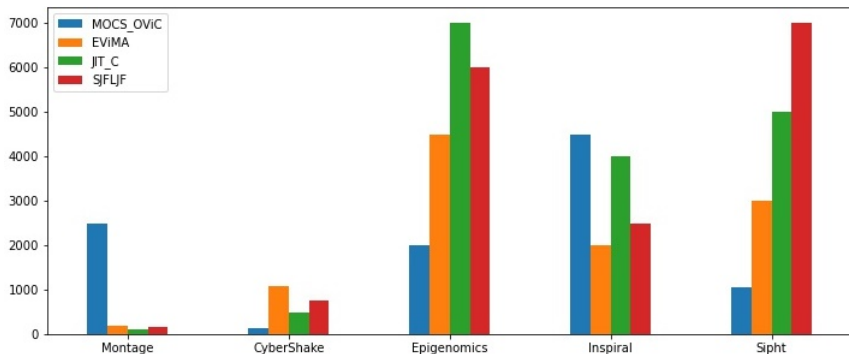


Figure – Le Makespan dans les workflow de petite taille

Les Workflows De Taille Moyenne

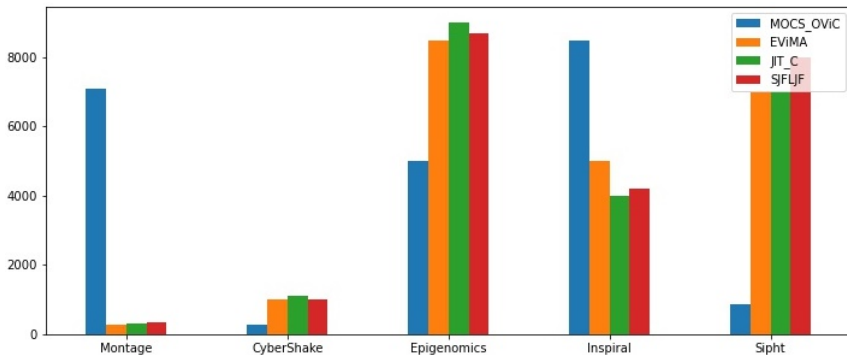


Figure – Le Makespan dans les workflow de taille moyenne

Les Workflows De Grande Taille

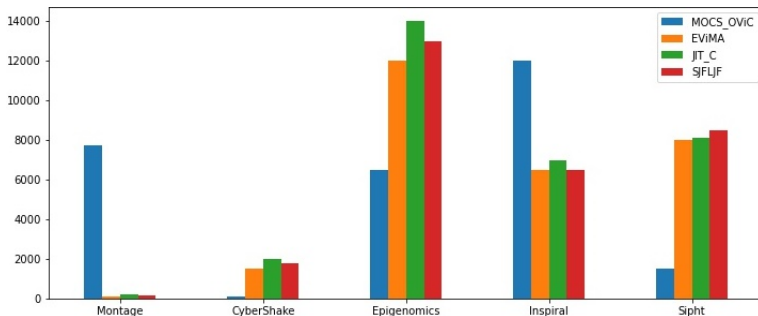


Figure – Le Makespan dans les workflow de grande taille

Les Workflows De Très Grande Taille

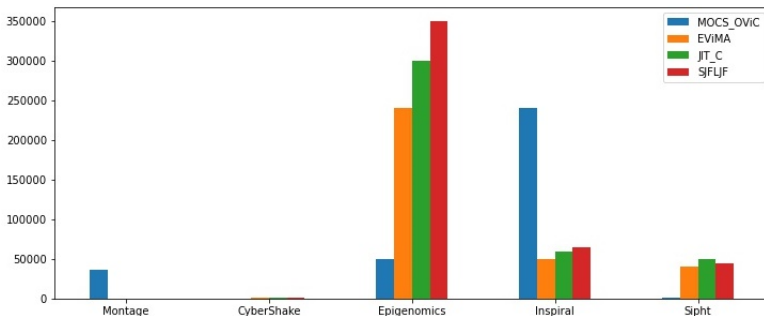


Figure – Le Makespan dans les workflows de très grande taille

Comparaison

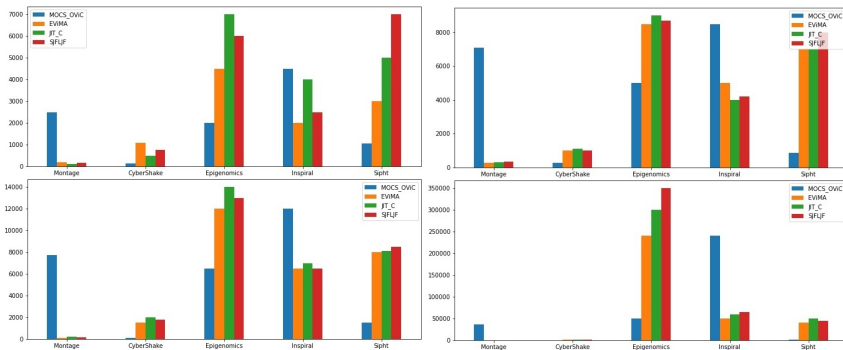


Figure – La comparaison du Makespan

Discussion

Avantage

Notre approche produit des plans d'exécution les plus courts pour CyberShake, Epigenomics et Sipht dans presque toutes les tailles parce que ces 3 workflows sont des applications gourmandes en CPU et MOCS-OViC équilibre bien les charges entre les hôtes.

Inconvénient

MOCS-OViC génère les pires plannings lorsque Montage et Inspiral sont exécutés. En effet, Inspiral nécessite beaucoup de mémoire, donc les migrations de VMs prennent plus de temps. Le montage est gourmand en transfert des données, MOCS-OViC ne considère pas le temps de transfert dans la génération des plans d'exécutions.

Les Workflows De Petite Taille

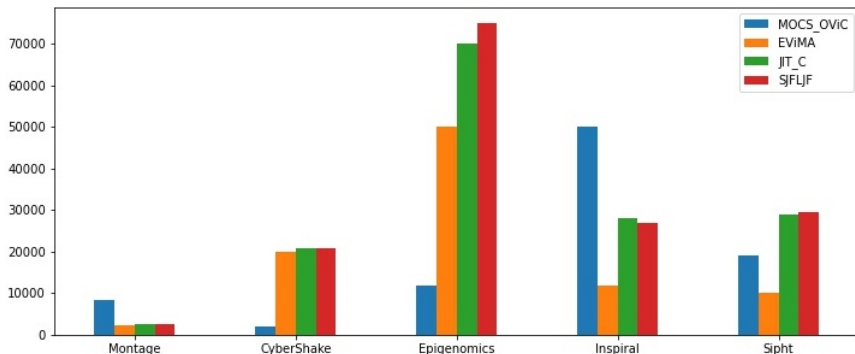


Figure – Le coût de l'exécution des workflow de petite taille

Les Workflows De Taille Moyenne

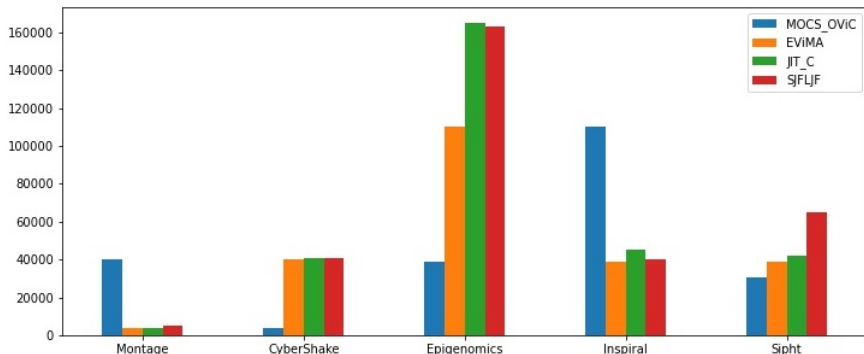


Figure – Le coût de l'exécution des workflow de taille moyenne

Les Workflows De Grande Taille

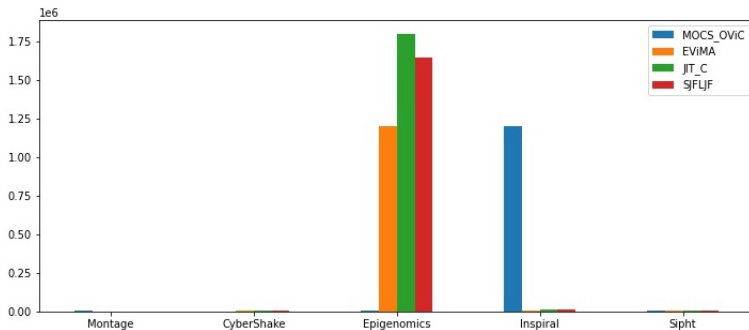


Figure – Le coût de l'exécution des workflow de grande taille

Les Workflows De Très Grande Taille

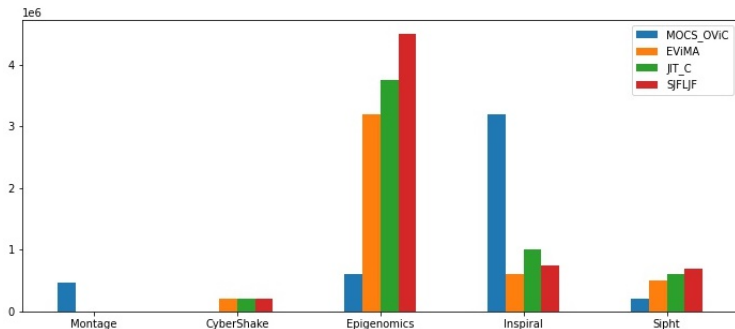


Figure – Le coût de l'exécution des workflows de très grande taille

Comparaison

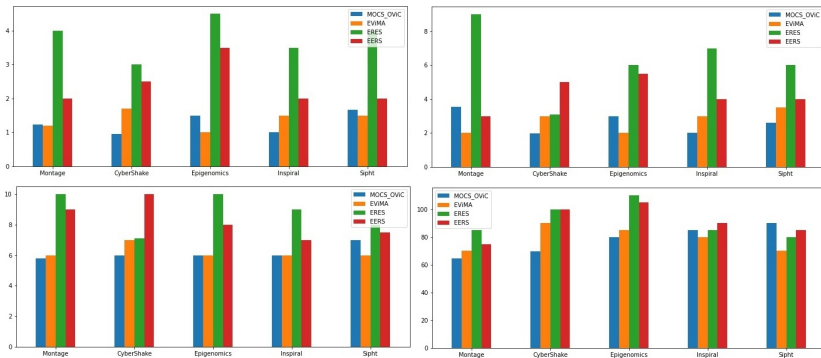


Figure – La comparaison des coûts de tous les algorithmes

Discussion

Avantages

Notre approche produit des solutions moins coûteuses pour CyberShake, Epigenomics et Sipht dans presque toutes les tailles. En effet, notre algorithme génère une meilleure durée ordonnancement, de sorte que le temps d'utilisation des machines virtuelles est moins.

Inconvénient

MOCS-OViC génère les pire solutions en termes de coût lorsque Montage et Inspiral sont exécutés. En effet, MOCS-OViC est un algorithme non sensible aux données et il génère un grand makespan, de sorte que l'utilisation des machines virtuelles est plus importante.

Conclusion

- Nombreux travaux de recherche étudient le problème de la planification des workflows scientifiques sur le Cloud.
- On a étudié dans ce travail le problème d'optimisation du coût, du makespan et de l'énergie.
- Nous proposons notre approche MOCS-OViC qui :
 - se base sur EViMA pour affecter les tâches aux VMs.
 - utilise certaines politiques de placement et de consolidation des VMs pour optimiser la consommation d'énergie :
 - ① détermine quelles VMs à migrer depuis la source PM et quand.
 - ② le PM de destination est sélectionné pour placer la VM sur celui-ci.

Conclusion

- Nombreux travaux de recherche étudient le problème de la planification des workflows scientifiques sur le Cloud.
- On a étudié dans ce travail le problème d'optimisation du coût, du makespan et de l'énergie.
- Nous proposons notre approche MOCS-OViC qui :
 - se base sur EViMA pour affecter les tâches aux VMs.
 - utilise certaines politiques de placement et de consolidation des VMs pour optimiser la consommation d'énergie :
 - 1 détermine quelles VMs à migrer depuis la source PM et quand.
 - 2 le PM de destination est sélectionné pour placer la VM sur celui-ci.

Conclusion

- Nombreux travaux de recherche étudient le problème de la planification des workflows scientifiques sur le Cloud.
- On a étudié dans ce travail le problème d'optimisation du coût, du makespan et de l'énergie.
- Nous proposons notre approche MOCS-OViC qui :
 - se base sur EViMA pour affecter les tâches aux VMs.
 - utilise certaines politiques de placement et de consolidation des VMs pour optimiser la consommation d'énergie :
 - 1 détermine quelles VMs à migrer depuis la source PM et quand.
 - 2 le PM de destination est sélectionné pour placer la VM sur celui-ci.

Conclusion

- Nous avons évalué notre algorithme :
 - en exécutant des expériences et en comparant les résultats avec certains algorithmes existants.
 - en utilisant 5 workflows scientifiques de différentes tailles.
- Les résultats sont satisfaisants lorsque des workflows intensifs en CPU sont exécutés.
- Le délai d'acquisition et de terminaison d'instance ne sont pas pris en compte.

Conclusion

- Nous avons évalué notre algorithme :
 - en exécutant des expériences et en comparant les résultats avec certains algorithmes existants.
 - en utilisant 5 workflows scientifiques de différentes tailles.
- Les résultats sont satisfaisants lorsque des workflows intensifs en CPU sont exécutés.
- Le délai d'acquisition et de terminaison d'instance ne sont pas pris en compte.

Conclusion

- Nous avons évalué notre algorithme :
 - en exécutant des expériences et en comparant les résultats avec certains algorithmes existants.
 - en utilisant 5 workflows scientifiques de différentes tailles.
- Les résultats sont satisfaisants lorsque des workflows intensifs en CPU sont exécutés.
- Le délai d'acquisition et de terminaison d'instance ne sont pas pris en compte.

Conclusion

- Nous avons l'intention d'appliquer ce travail dans un environnement Fog Computing :
 - réduire la consommation d'énergie et les coûts opérationnels et l'impact environnemental.
 - réduire du makespan pour améliorer QoS et l'expérience utilisateur des applications Fog.
 - améliorer la fiabilité et la scalabilité et éviter la congestion du réseau.

Références



So many clouds-what's the difference ?, May 2020.



Katharina Görlach, Mirko Sonntag, Dimka Karastoyanova, Frank Leymann, and Michael Reiter.

Conventional workflow technology for scientific simulation.

Guide to e-Science : Next Generation Scientific Research and Discovery, pages 323–352, 2011.



Hachem Guerid.

Introduction au cloud computing, 3 2022.



Mandeep Kaur and Rajni Aron.

An energy-efficient load balancing approach for scientific workflows in fog computing.

Wireless Personal Communications, 125(4), 2022