Résister à la performance maximale ?

Anne Benoit et Yves Robert
LIP, Ecole Normale Supérieure de Lyon
Institut Universitaire de France

Congrès IUF – Résistance(s)

Motivation

- Calcul haute performance (**HPC**: *High-performance Computing*) – beaucoup d'applications très gourmandes en calcul, qui tournent sur des supercalculateurs
- Quête de performance: faire le plus possible d'opérations par seconde
- Impératif économique, facilité et qualité du service des centres de calcul

- Boom de l'IA, course aux *publications dénuées de sens*
- Aller vite, toujours plus vite

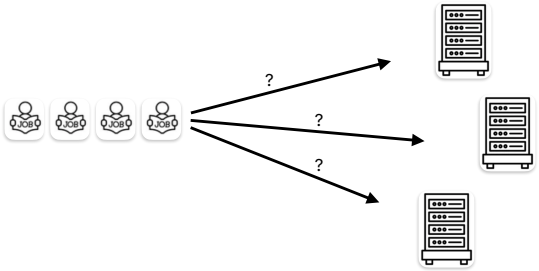
Motivation

- Calcul haute performance (**HPC**: *High-performance Computing*) – beaucoup d'applications très gourmandes en calcul, qui tournent sur des supercalculateurs
- Quête de performance: faire le plus possible d'opérations par seconde
- Impératif économique, facilité et qualité du service des centres de calcul

- Boom de l'IA, course aux *publications dénuées de sens*
- Aller vite, toujours plus vite

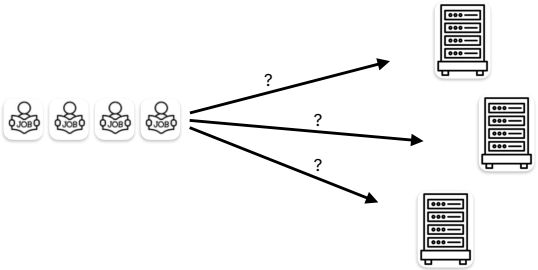
Motivation

- Les utilisateurs soumettent des jobs
- Le système décide où les envoyer pour les exécuter
- Quel est l'objectif ?
  - Aller vite, toujours plus vite ?
  - Ne pas faire attendre un utilisateur trop longtemps ?
  - Ne pas consommer trop d'énergie ?



Motivation

- Les utilisateurs soumettent des jobs
- Le système décide où les envoyer pour les exécuter
- Quel est l'objectif ?
  - Aller vite, toujours plus vite ?
  - Ne pas faire attendre un utilisateur trop longtemps ?
  - Ne pas consommer trop d'énergie ?

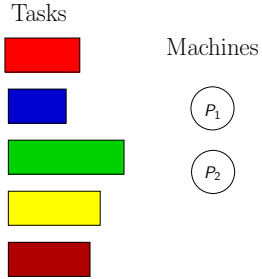


Problèmes d'ordonnancement

Allouer des ressources à des applications pour optimiser des métriques de performance

- Ressources: Systèmes distribués à grande échelle, avec des millions de composants
- Applications: Applications parallèles, représentées par un ensemble de tâches, ou alors par une quantité de travail à accomplir
- Métriques de performance: La principale est la performance de l'application, à savoir aller le plus vite possible à tout prix!

Problèmes d'ordonnancement classiques



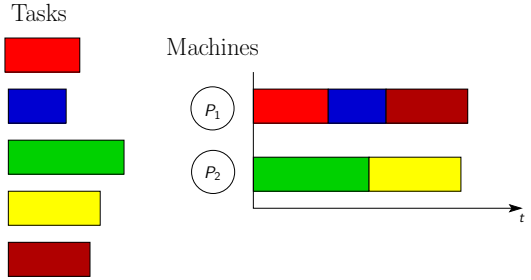
Objectifs:

- Minimiser le temps total d'exécution (C_{max})
- Minimiser la somme pondérée des temps d'exécution $\sum_i w_i C_i$

Résultats:

- Algorithmes optimaux, ou résultats de NP-complétude, i.e., montrant la difficulté intrinsèque du problème

Problèmes d'ordonnancement classiques



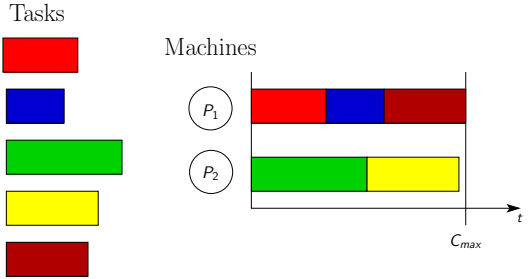
Objectifs:

- Minimiser le temps total d'exécution (C_{max})
- Minimiser la somme pondérée des temps d'exécution $\sum_i w_i C_i$

Résultats:

- Algorithmes optimaux, ou résultats de NP-complétude, i.e., montrant la difficulté intrinsèque du problème

Problèmes d'ordonnancement classiques



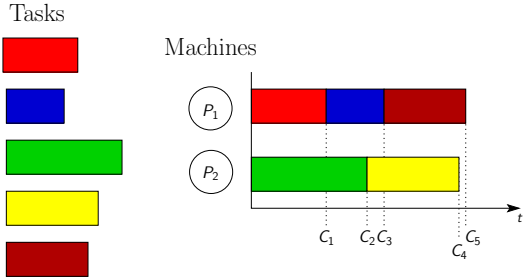
Objectifs:

- Minimiser le temps total d'exécution (C_{max})
- Minimiser la somme pondérée des temps d'exécution $\sum_i w_i C_i$

Résultats:

- Algorithmes optimaux, ou résultats de NP-complétude, i.e., montrant la difficulté intrinsèque du problème

Problèmes d'ordonnancement classiques



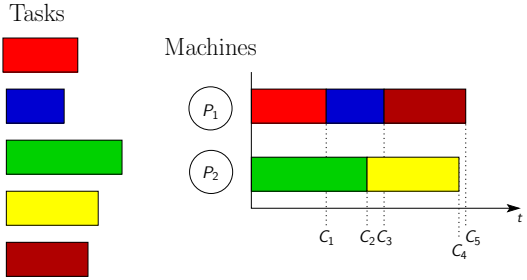
Objectifs:

- Minimiser le temps total d'exécution (C_{max})
- Minimiser la somme pondérée des temps d'exécution $\sum_i w_i C_i$

Résultats:

- Algorithmes optimaux, ou résultats de NP-complétude, i.e., montrant la difficulté intrinsèque du problème

Problèmes d'ordonnancement classiques



Objectifs:

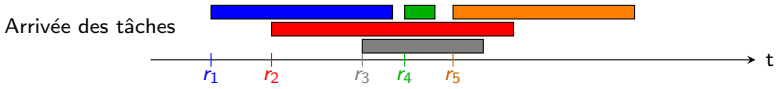
- Minimiser le temps total d'exécution (C_{max})
- Minimiser la somme pondérée des temps d'exécution $\sum_i w_i C_i$

Résultats:

- Algorithmes optimaux, ou résultats de NP-complétude, i.e., montrant la difficulté intrinsèque du problème

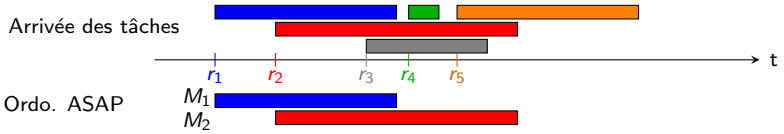
Ordonnancement en ligne

- Techniques d'ordonnancement en ligne: décider où et quand exécuter les tâches sur des ressources – au coeur des *batch schedulers*
- Problème de base: ordonnancer des tâches indépendantes sur des plateformes de HPC parallèles
- **Fonctions objectif:**
  - Utilisation (point de vue plateforme) – fraction de temps pendant lequel la plateforme calcule
  - Stretch (point de vue utilisateur) – minimiser le stretch maximum (ou moyen) des tâches, i.e., leur temps de réponse normalisé par leur longueur



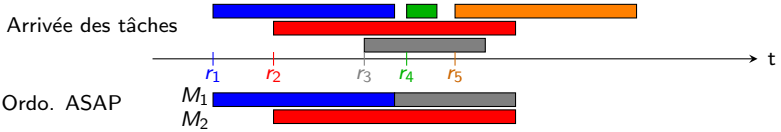
Ordonnancement en ligne

- Techniques d'ordonnancement en ligne: décider où et quand exécuter les tâches sur des ressources – au coeur des *batch schedulers*
- Problème de base: ordonnancer des tâches indépendantes sur des plateformes de HPC parallèles
- **Fonctions objectif:**
  - Utilisation (point de vue plateforme) – fraction de temps pendant lequel la plateforme calcule
  - Stretch (point de vue utilisateur) – minimiser le stretch maximum (ou moyen) des tâches, i.e., leur temps de réponse normalisé par leur longueur



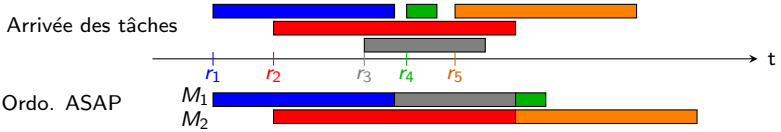
Ordonnancement en ligne

- Techniques d'ordonnancement en ligne: décider où et quand exécuter les tâches sur des ressources – au coeur des *batch schedulers*
- Problème de base: ordonnancer des tâches indépendantes sur des plateformes de HPC parallèles
- **Fonctions objectif:**
  - Utilisation (point de vue plateforme) – fraction de temps pendant lequel la plateforme calcule
  - Stretch (point de vue utilisateur) – minimiser le stretch maximum (ou moyen) des tâches, i.e., leur temps de réponse normalisé par leur longueur



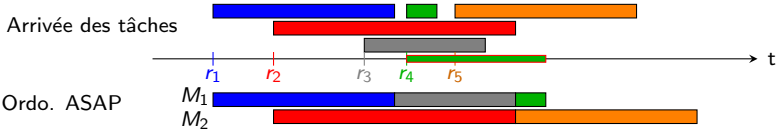
Ordonnancement en ligne

- Techniques d'ordonnancement en ligne: décider où et quand exécuter les tâches sur des ressources – au coeur des *batch schedulers*
- Problème de base: ordonnancer des tâches indépendantes sur des plateformes de HPC parallèles
- **Fonctions objectif:**
  - Utilisation (point de vue plateforme) – fraction de temps pendant lequel la plateforme calcule
  - Stretch (point de vue utilisateur) – minimiser le stretch maximum (ou moyen) des tâches, i.e., leur temps de réponse normalisé par leur longueur



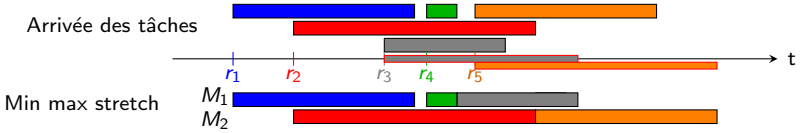
Ordonnancement en ligne

- Techniques d'ordonnancement en ligne: décider où et quand exécuter les tâches sur des ressources – au coeur des *batch schedulers*
- Problème de base: ordonnancer des tâches indépendantes sur des plateformes de HPC parallèles
- **Fonctions objectif:**
  - Utilisation (point de vue plateforme) – fraction de temps pendant lequel la plateforme calcule
  - Stretch (point de vue utilisateur) – minimiser le stretch maximum (ou moyen) des tâches, i.e., leur temps de réponse normalisé par leur longueur



Ordonnancement en ligne

- Techniques d'ordonnancement en ligne: décider où et quand exécuter les tâches sur des ressources – au coeur des *batch schedulers*
- Problème de base: ordonnancer des tâches indépendantes sur des plateformes de HPC parallèles
- **Fonctions objectif:**
  - Utilisation (point de vue plateforme) – fraction de temps pendant lequel la plateforme calcule
  - Stretch (point de vue utilisateur) – minimiser le stretch maximum (ou moyen) des tâches, i.e., leur temps de réponse normalisé par leur longueur



Faut-il résister à cette quête de performance ?

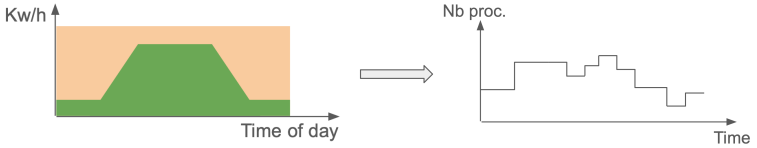
- Effet rebond: plus on a de puissance de calcul, plus on l'utilise
- Utiliser à bon escient, par exemple pour des simulations en aérodynamique, des applications en santé
- Une façon de **résister** consiste à utiliser l'énergie d'une meilleure façon:
  - Utiliser le centre de calcul lorsqu'il est alimenté en énergie renouvelable → limiter les émissions de CO2
  - Applications non critiques: Accepter un temps de réponse légèrement plus grand pour des gains d'énergie (compromis performance/énergie)
  - Accepter des compromis en dialoguant avec le "batch scheduler": Faire tourner l'application sur moins de processeurs qu'initialement demandé

Plan

Nouveau défi: Capacité variable

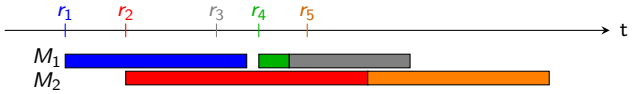
- Les centres de calcul actuels supposent que les ressources sont en nombre fixe:

- Approches émergentes ⇒ Puissance variable
  - Exploiter l'énergie renouvelable
  - Réduire les émissions carbone



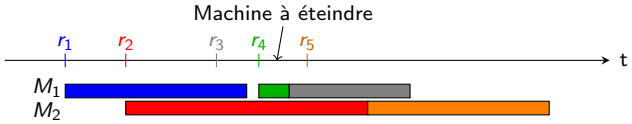
Ordonnancement à capacité variable

- Green computing: la puissance disponible évolue dans le temps (énergie solaire ou vent, etc.)
- Comment ordonnancer efficacement quand la puissance disponible varie, i.e., le nombre de machines qui peuvent être alimentées varie dans le temps ?
- Besoin d'être prêt pour ces variations: si une machine est éteinte, il faut ré-exécuter ses tâches, et démarrer de nouvelles tâches quand une nouvelle machine est allumée



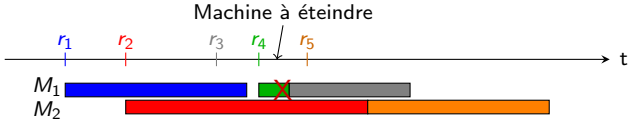
Ordonnancement à capacité variable

- Green computing: la puissance disponible évolue dans le temps (énergie solaire ou vent, etc.)
- Comment ordonnancer efficacement quand la puissance disponible varie, i.e., le nombre de machines qui peuvent être alimentées varie dans le temps ?
- Besoin d'être prêt pour ces variations: si une machine est éteinte, il faut ré-exécuter ses tâches, et démarrer de nouvelles tâches quand une nouvelle machine est allumée



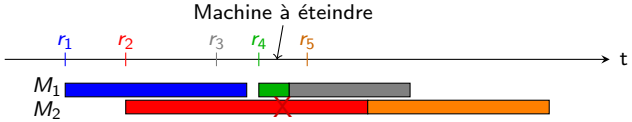
Ordonnancement à capacité variable

- Green computing: la puissance disponible évolue dans le temps (énergie solaire ou vent, etc.)
- Comment ordonnancer efficacement quand la puissance disponible varie, i.e., le nombre de machines qui peuvent être alimentées varie dans le temps ?
- Besoin d'être prêt pour ces variations: si une machine est éteinte, il faut ré-exécuter ses tâches, et démarrer de nouvelles tâches quand une nouvelle machine est allumée



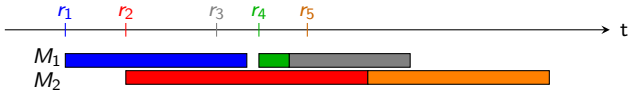
Ordonnancement à capacité variable

- Green computing: la puissance disponible évolue dans le temps (énergie solaire ou vent, etc.)
- Comment ordonnancer efficacement quand la puissance disponible varie, i.e., le nombre de machines qui peuvent être alimentées varie dans le temps ?
- Besoin d'être prêt pour ces variations: si une machine est éteinte, il faut ré-exécuter ses tâches, et démarrer de nouvelles tâches quand une nouvelle machine est allumée



Conscient du risque ?

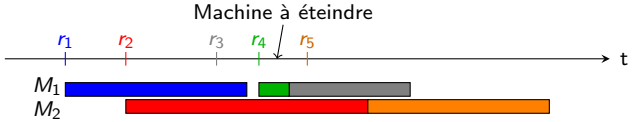
1. **Quelle machine éteindre ?**



2. **Comment ordonnancer les tâches pour minimiser l'impact ?**

Conscient du risque ?

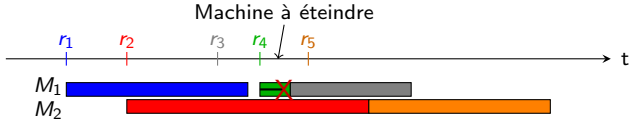
1. **Quelle machine éteindre ?**



2. **Comment ordonnancer les tâches pour minimiser l'impact ?**

Conscient du risque ?

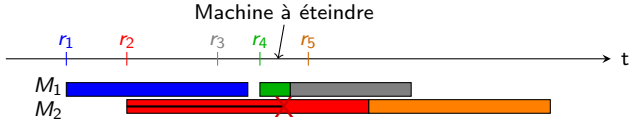
1. **Quelle machine éteindre ?**



2. **Comment ordonnancer les tâches pour minimiser l'impact ?**

Conscient du risque ?

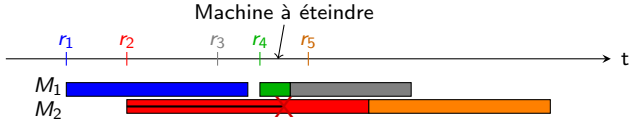
1. **Quelle machine éteindre ?**



2. **Comment ordonnancer les tâches pour minimiser l'impact ?**

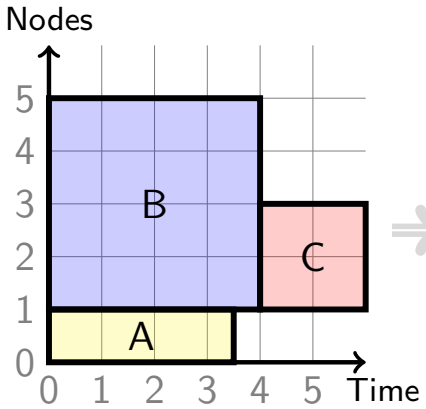
Conscient du risque ?

1. **Quelle machine éteindre ?**

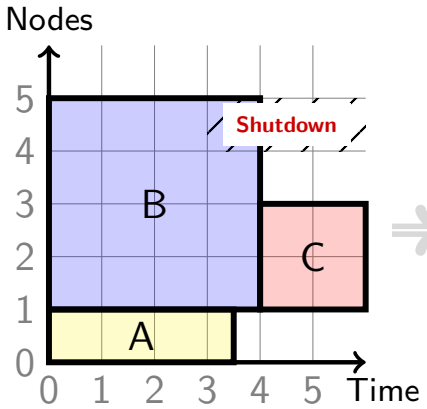


2. **Comment ordonnancer les tâches pour minimiser l'impact ?**

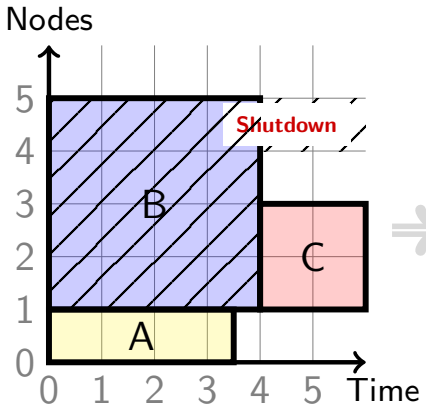
Un petit exemple



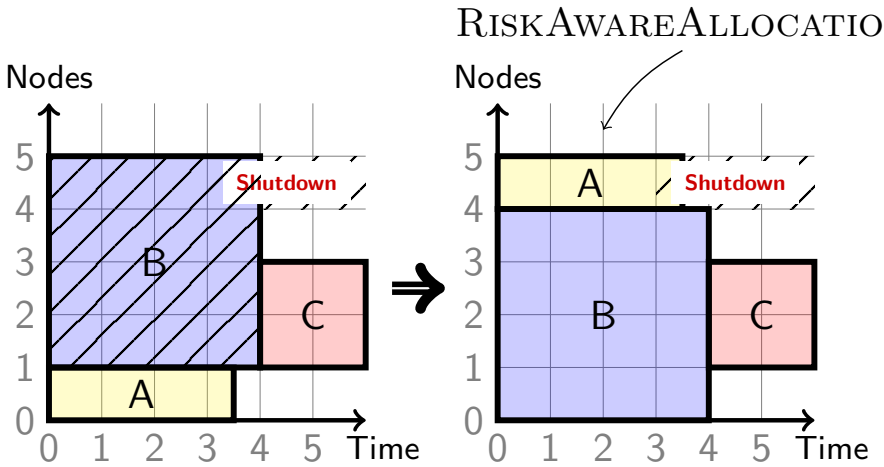
Un petit exemple



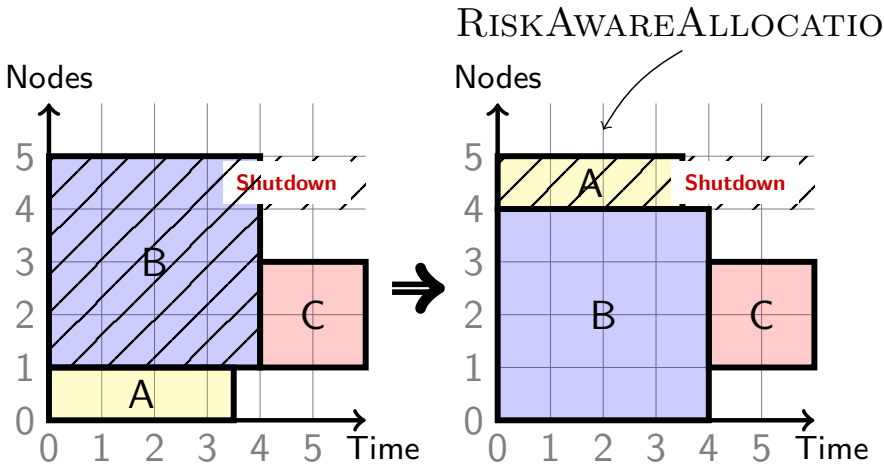
Un petit exemple



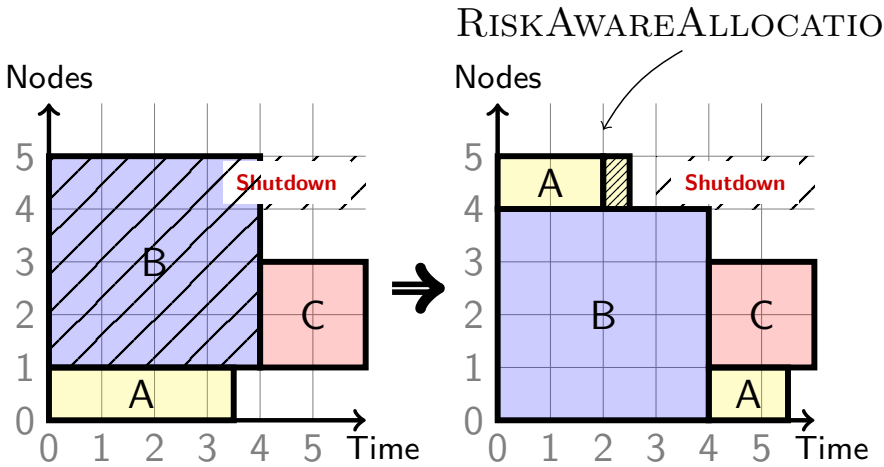
Un petit exemple



Un petit exemple



Un petit exemple



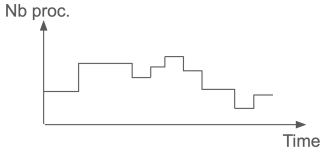
Principales questions

- Quand la puissance diminue, quelles machines éteindre ? Quelles tâches sont interrompues ? Et ré-ordonnancées ?
- Est-on prévenu en avance d'un changement de puissance ?
  - Les variations de ressources sont dictées par un mélange de disponibilité technique et de conditions économiques
  - Prédictions parfaites fournies par un oracle ? Peut-être trop optimiste ☹
- Ré-ordonnancer les tâches interrompues
  - Peut-on prendre un checkpoint proactif avant l'interruption ?
  - Quelle priorité donner à chaque tâche interrompue ?
  - Quelle géométrie et quelles machines pour la ré-exécution ?

Plan

Modèle

- Pas possible de checkpointer les jobs ou d'anticiper les variations de ressource

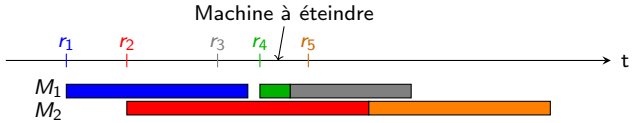


- Ensemble de jobs/tâches, à exécuter sur des machines multicoeurs identiques
- Le nombre de machines disponibles évolue dans le temps, nombre non connu en avance

Conception de stratégies qui prennent en compte le risque de devoir éteindre des machines,
allouant les nouvelles tâches à la *bonne* machine cible,
suivant le critère à optimiser

Fonctions objectif

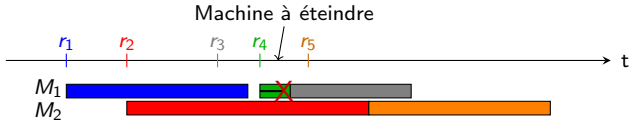
- Utilisation de la plateforme: Ce n'est plus un bon critère, certaines tâches peuvent être interrompues et du travail perdu...



- **Fonctions objectif:**
  - Goodput ⇒ utilisation *effective*, ne compte que le travail *utile*
  - Stretch pour l'équité entre utilisateurs

Fonctions objectif

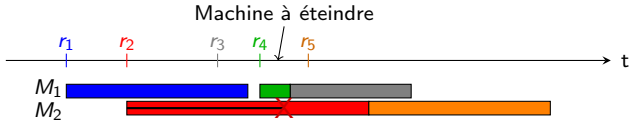
- Utilisation de la plateforme: Ce n'est plus un bon critère, certaines tâches peuvent être interrompues et du travail perdu...



- **Fonctions objectif:**
  - Goodput ⇒ utilisation *effective*, ne compte que le travail *utile*
  - Stretch pour l'équité entre utilisateurs

Fonctions objectif

- Utilisation de la plateforme: Ce n'est plus un bon critère, certaines tâches peuvent être interrompues et du travail perdu...

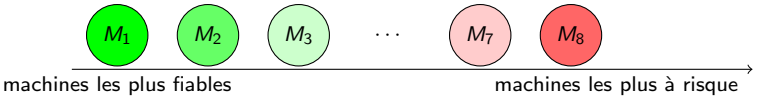


- **Fonctions objectif:**
  - Goodput ⇒ utilisation *effective*, ne compte que le travail *utile*
  - Stretch pour l'équité entre utilisateurs

Algorithmes

Idées principales:

- Prise de décision à chaque événement (arrivée ou fin de tâche, ajout ou suppression de machine)
- Ordonner les machines pour permettre un choix guidé:



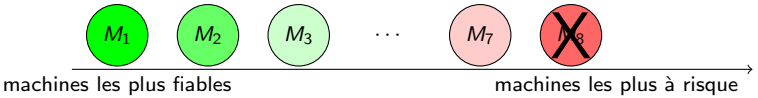
Stratégies d'allocation des tâches conscientes du risque

- Le plus naturel: first-fit, remplir d'abord les machines les plus fiables
- Pour perdre moins de travail, placer les grandes tâches sur les machines fiables !

Algorithmes

Idées principales:

- Prise de décision à chaque événement (arrivée ou fin de tâche, ajout ou suppression de machine)
- Ordonner les machines pour permettre un choix guidé:



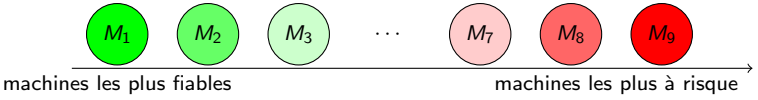
Stratégies d'allocation des tâches conscientes du risque

- Le plus naturel: first-fit, remplir d'abord les machines les plus fiables
- Pour perdre moins de travail, placer les grandes tâches sur les machines fiables !

Algorithmes

Idées principales:

- Prise de décision à chaque événement (arrivée ou fin de tâche, ajout ou suppression de machine)
- Ordonner les machines pour permettre un choix guidé:



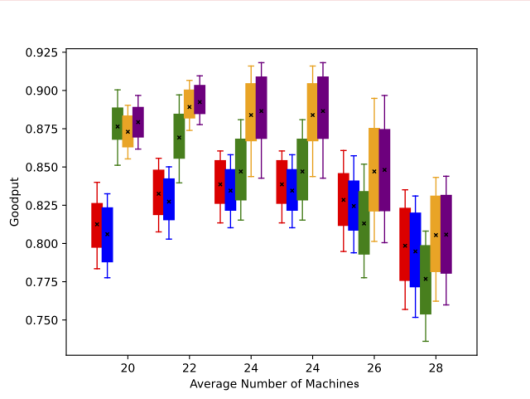
Stratégies d'allocation des tâches conscientes du risque

- Le plus naturel: first-fit, remplir d'abord les machines les plus fiables
- Pour perdre moins de travail, placer les grandes tâches sur les machines fiables !

Algorithmes

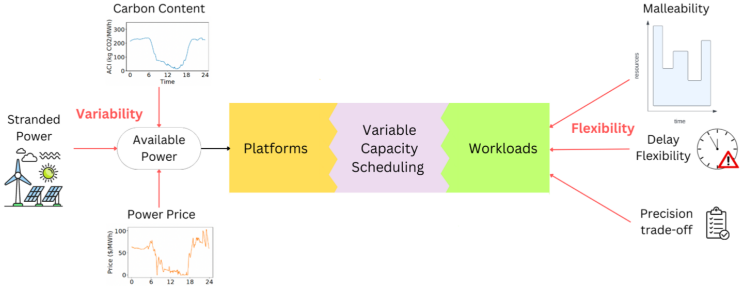
Résultats de simulation:

Gains significatifs en plaçant la bonne tâche sur la bonne machine



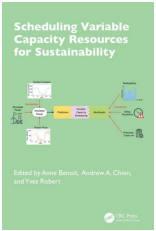
Plan

Conclusion



Nombreux problèmes d'ordonnancement motivants lorsque le nombre de ressources change dans le temps ☺

Rapports de workshops: *Scheduling Variable Capacity Resources for Sustainability*; Mars 2023, et septembre 2025, U. Chicago Paris Center



Conclusion

Résister à la performance maximale

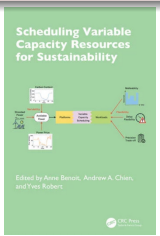
S'intéresser à d'autres critères d'optimisation

Persuader les utilisateurs d'accepter un délai avant le lancement de leur job

Approche soucieuse de compromis environnementaux et sociétaux

Nombreux problèmes d'ordonnancement motivants lorsque le nombre de ressources change dans le temps ☺

Rapports de workshops: *Scheduling Variable Capacity Resources for Sustainability*; Mars 2023, et septembre 2025, U. Chicago Paris Center



Références

- L. Perotin, C. Zhang, R. Wijayawardana, A. Benoit, Y. Robert, A. Chien. **Risk-Aware Scheduling Algorithms for Variable Capacity Resources**. In Proceedings of PMBS, in conjunction with SC'23, Denver, USA, November 2023.
- J. Cendrier, A. Benoit, F. Vivien. **Scheduling Jobs Under a Variable Number of Processors**. IEEE Transactions on Parallel and Distributed Systems, vol. 37, pp. 427-442, February 2026.
- A. Benoit, A. Chien, Y. Robert, editors. **Scheduling Variable Capacity Resources for Sustainability**. Chapman and Hall/CRC Press 2026.