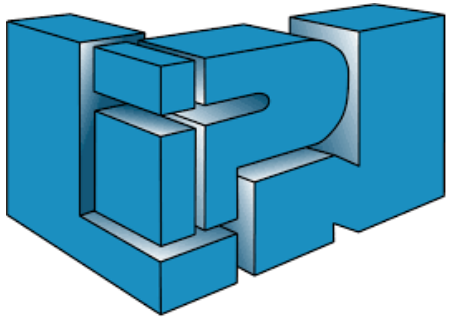
Système d'allocation de ressources combiné avec un modèle économique

Tarek Menouer
Laboratoire d'Informatique de Paris Nord

5 Juillet 2017



Outline

Outline

Introduction

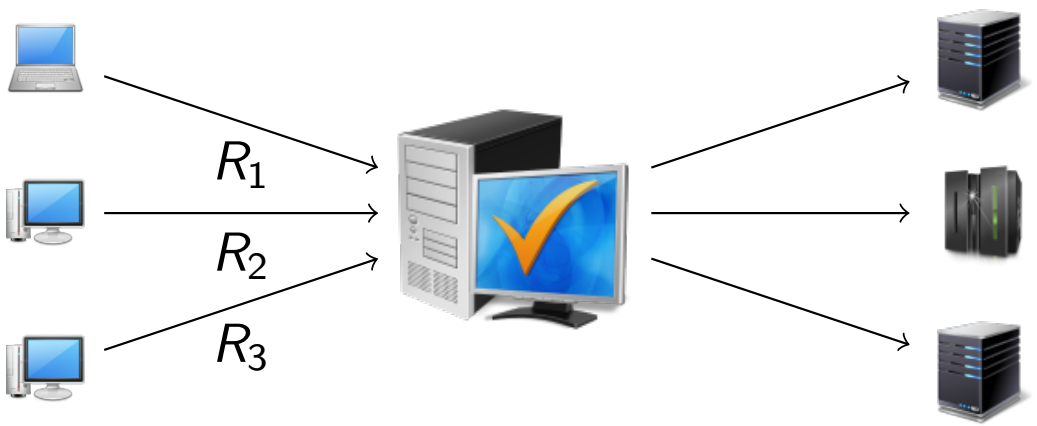
Motivation

- Répondre aux problèmes posés par un service d'exécution en mode SaaS :
  - Chaque requête représente un service qui s'exécute en utilisant un ou plusieurs cœurs de calcul
  - Infrastructure *cloud computing* réservée et fixée
- Proposer un système d'allocation de ressources
  - Optimiser l'allocation de plusieurs requêtes soumises en ligne

Introduction

Motivation

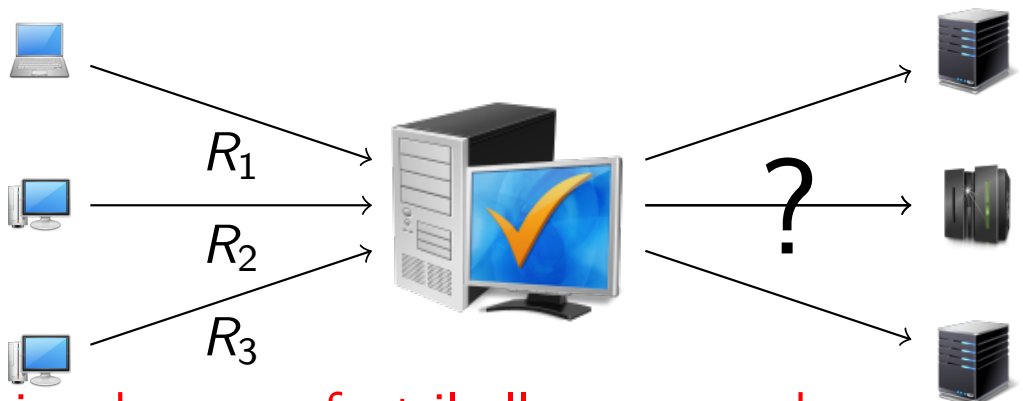
- Répondre aux problèmes posés par un service d'exécution en mode SaaS :
  - Chaque requête représente un service qui s'exécute en utilisant un ou plusieurs cœurs de calcul
  - Infrastructure *cloud computing* réservée et fixée
- Proposer un système d'allocation de ressources
  - Optimiser l'allocation de plusieurs requêtes soumises en ligne



Introduction

Motivation

- Répondre aux problèmes posés par un service d'exécution en mode SaaS :
  - Chaque requête représente un service qui s'exécute en utilisant un ou plusieurs cœurs de calcul
  - Infrastructure *cloud computing* réservée et fixée
- Proposer un système d'allocation de ressources
  - Optimiser l'allocation de plusieurs requêtes soumises en ligne



Combien de cœurs faut-il allouer pour chaque requête ?

Introduction

Dans la littérature

- Plusieurs systèmes d'allocation de ressources existent pour les Clusters/Grid/Cloud :
  - PBS, LoadLeveler, Maui, Torque, Globus, ...

✗ S'adresse aux connaisseurs qui savent combien de cœurs il faut allouer pour chaque requête

Dans notre étude

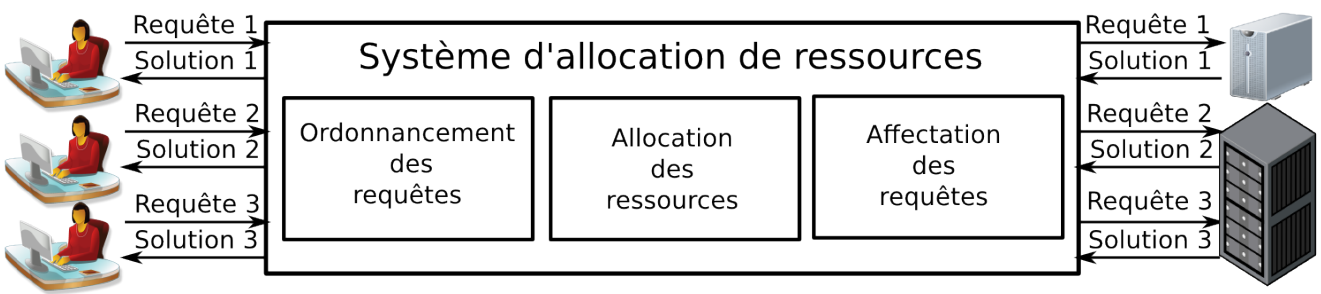
- Le client n'a pas de connaissances exactes sur la parallélisation des applications
- Nombre de cœurs affectés à chaque application :
  - Modèle économique
  - La charge des machines dans l'infrastructure
- Privilégier les clients selon leurs modèles économiques

Outline

Système d'allocation de ressources basé sur un modèle économique (1)

Objectifs

- Adapter le temps d'attente des clients selon leurs modèles économiques
- Partager le nombre de cœurs de calcul selon le modèle économique des clients et la charge des machines dans l'infrastructure
- Affecter une requête à la machine qui réduit le coût de l'infrastructure
- Optimiser l'ordonnancement de toutes les requêtes



Modèle économique

Classe SLA

- 2 classes SLAs :
  - Classe qualitative : Le temps de début d'exécution des requêtes
  - Classe quantitative : Le nombre de cœurs de calcul

Services pour chaque classe SLA

- Premium : Exécuter une requête le plus tôt possible (qualitative) ou allouer un grand nombre de cœurs (quantitative)
- Advanced : Exécuter une requête pendant un temps réduit (qualitative) ou allouer un nombre moyen de cœurs (quantitative)
- Best effort : Exécuter une requête sans contrainte sur le temps de début d'exécution (qualitative) ni sur le nombre de cœurs (quantitative)
- Chaque service a une priorité :
  - Exemple : Priorité 3 pour Premium, 2 pour Advanced et 1 pour Best effort

Classe SLA quantitative

Hypothèses

- Chaque application s'exécute au plus sur une seule machine

Répartition des cœurs selon le service

- Limiter le nombre de cœurs pour chaque service
- K= le nombre de cœurs de la machine la plus petite
  - Premium
    - Max Cœurs = K
    - Min Cœurs = $0.67 \times K$
  - Advanced
    - Max Cœurs = $0.66 \times K$
    - Min Cœurs = $0.34 \times K$
  - Best effort
    - Max Cœurs = $0.33 \times K$
    - Min Cœurs = 1

Ordonnancement des requêtes

Principe

- Insérer chaque nouvelle requête $\Rightarrow$ une file globale
- Sélectionner la première requête selon le retour de l'algorithme PROMETHEE II (*Preference Ranking Organisation METHod for Enrichment Evaluations*)

Algorithme PROMETHEE II

- Méthode multi-critères d'aide à la décision utilisée en :

Ordonnancement des requêtes

Principe

- Insérer chaque nouvelle requête $\Rightarrow$ une file globale
- Sélectionner la première requête selon le retour de l'algorithme PROMETHEE II (*Preference Ranking Organisation METHod for Enrichment Evaluations*)

Algorithme PROMETHEE II

- Méthode multi-critères d'aide à la décision utilisée en :

Environment Management, Hydrology and Water Management, Business and Financial Management, Chemistry, Logistics and Transportation, Manufacturing and Assembly, Energy Management, Social, Medicine, Agriculture, Education, Design, Government and Sports.

Ordonnancement des requêtes

Principe

- Insérer chaque nouvelle requête ⇒ une file globale
- Sélectionner la première requête selon le retour de l'algorithme PROMETHEE II (*Preference Ranking Organisation METHod for Enrichment Evaluations*)

Algorithme PROMETHEE II

- Méthode multi-critères d'aide à la décision utilisée en :

  Environment Management, Hydrology and Water Management, Business and Financial Management, Chemistry, Logistics and Transportation, Manufacturing and Assembly, Energy Management, Social, Medicine, Agriculture, Education, Design, Government and Sports.

- Mathématiquement bien posé et socio-économiquement bien posé (combinaison de critères et seuils de perception)
- Elle donne un « bon compromis » entre le critère qualitatif et quantitatif

Allocation de ressources

Détermination de nombre des cœurs

- $R_1, \cdots, R_n$ représentent l'ensemble des requêtes sauvegardées dans la file avec leurs priorités $((P_{1,1}, P_{1,2}), (P_{2,1}, P_{2,2}), \cdots, (P_{n,1}, P_{n,2}))$
- T le nombre de cœurs inactifs dans toute l'infrastructure
- C_i le nombre de cœurs qui doit être alloué à la requête R_i qui a la priorité P_i

$$C_i = \frac{P_{i,2} * T}{\sum_{j=0}^{n} P_{j,2}}$$

Adaptation de nombre de cœurs

- Si $C_i >$ Max cœurs $\rightarrow C_i =$Max cœurs
- Si $C_i <$ Min cœurs $\rightarrow C_i =$Min cœurs

Affectation des requêtes

Principe

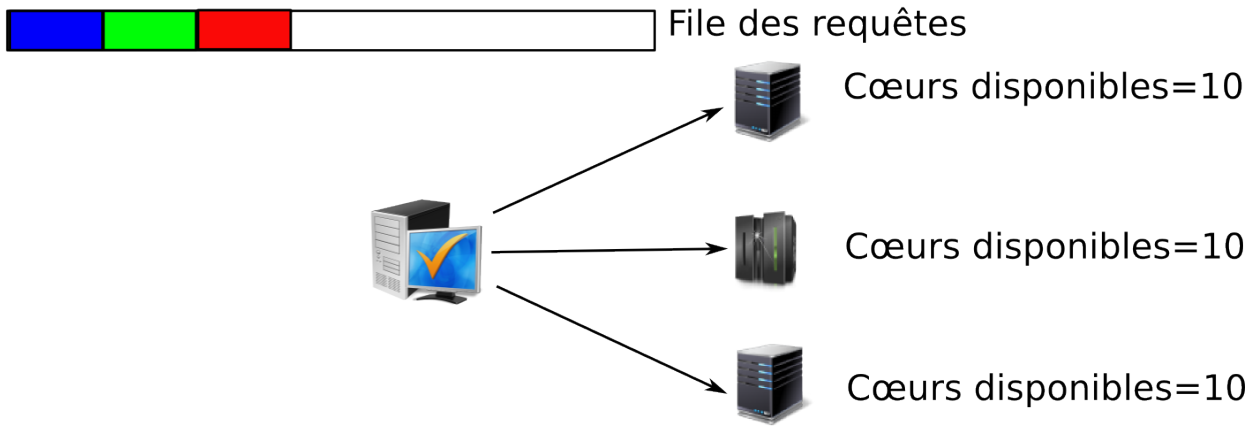
- Minimiser le nombre des machines actives afin de réduire le coût de l'infrastructure
- Affecter la requête sélectionnée (R_i) à la machine retournée par l'heuristique Bin Packing
  - La machine qui a le moins nombre de cœurs inactifs $\geq$ au nombre de cœurs calculé pour la requête R_i
- S'il n'y a pas une machine qui peut exécuter la requête (R_i), R_i est mise en attente
  - Si la requête (R_i) est mise en attente plusieurs fois, le système est bloqué jusqu'à ce que la requête R_i est exécutée afin d'éviter le problème de famine (starvation)

Exemple

Requête 1 : Premium service avec priorité = 3 dans les 2 classes SLAs, Min=5 et Max=6

Requête 2 : Advanced service avec priorité = 2 dans les 2 classes SLAs, Min=3 et Max=4

Requête 3 : Best effort service avec priorité = 1 dans les 2 classes SLAs, Min=1 et Max=2

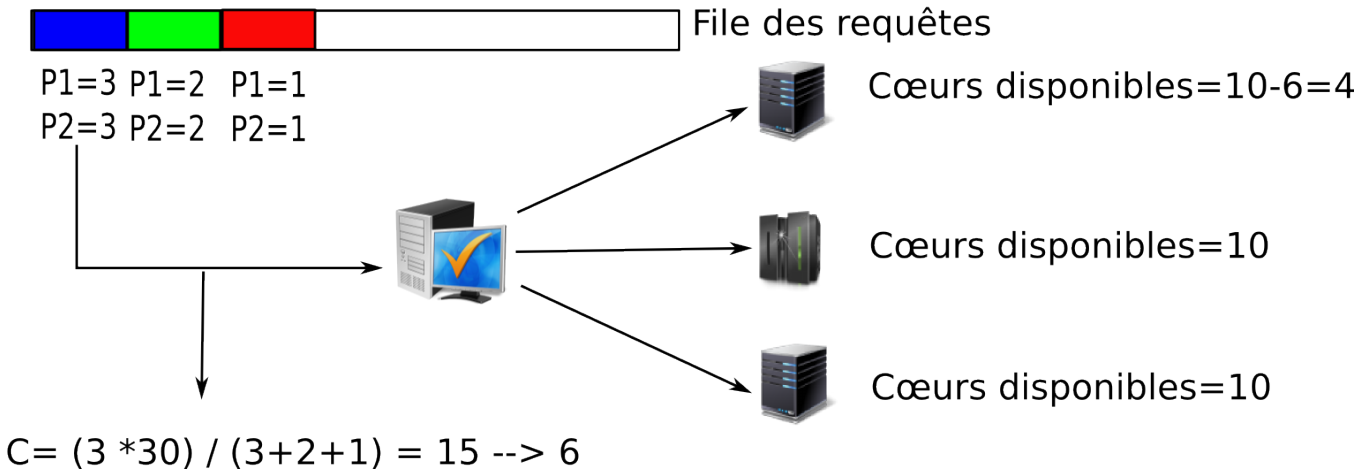


Exemple

Requête 1 : Premium service avec priorité = 3 dans les 2 classes SLAs, Min=5 et Max=6

Requête 2 : Advanced service avec priorité = 2 dans les 2 classes SLAs, Min=3 et Max=4

Requête 3 : Best effort service avec priorité = 1 dans les 2 classes SLAs, Min=1 et Max=2

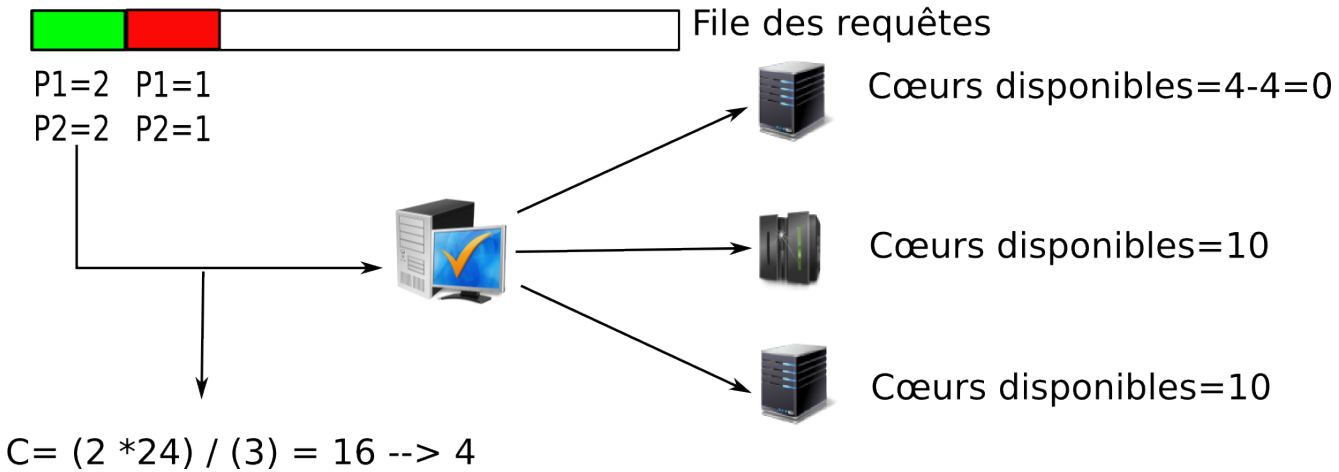


Exemple

Requête 1 : Premium service avec priorité = 3 dans les 2 classes SLAs, Min=5 et Max=6

Requête 2 : Advanced service avec priorité = 2 dans les 2 classes SLAs, Min=3 et Max=4

Requête 3 : Best effort service avec priorité = 1 dans les 2 classes SLAs, Min=1 et Max=2

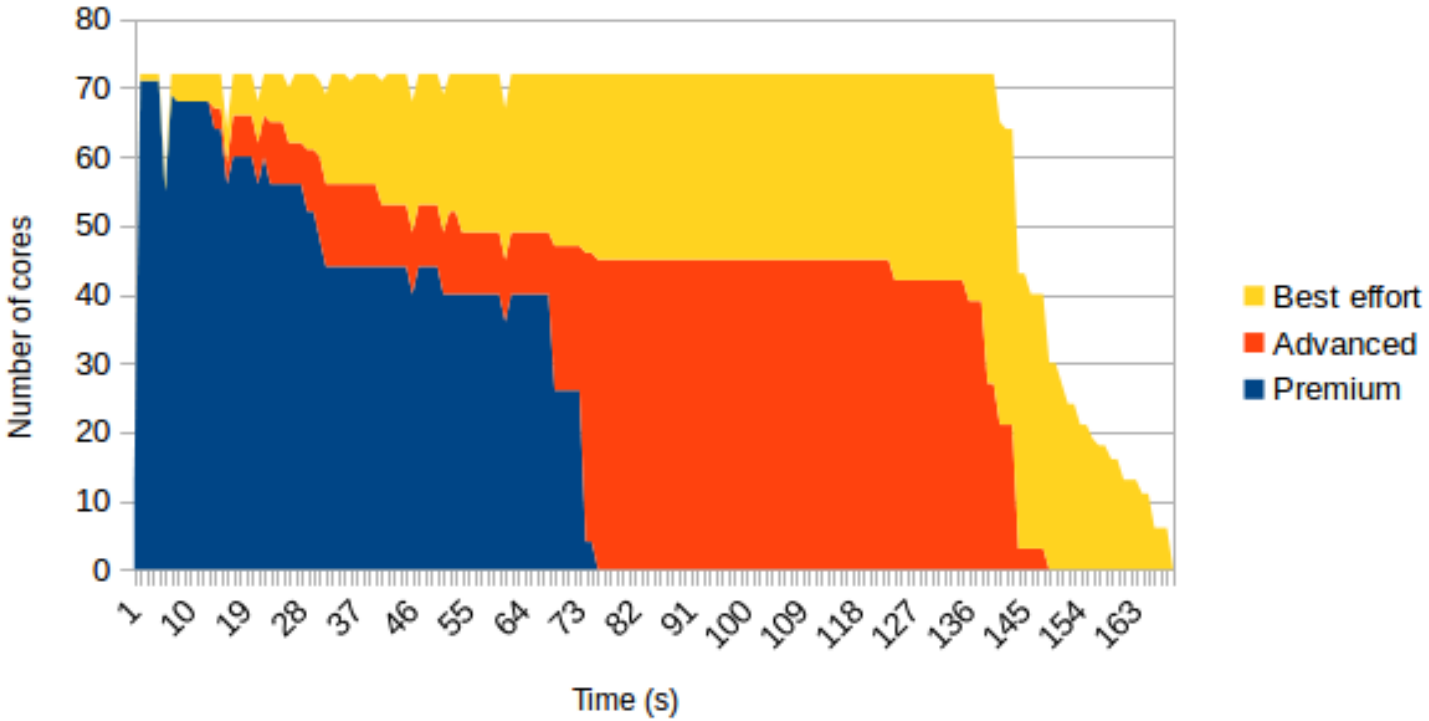


Outline

Protocole

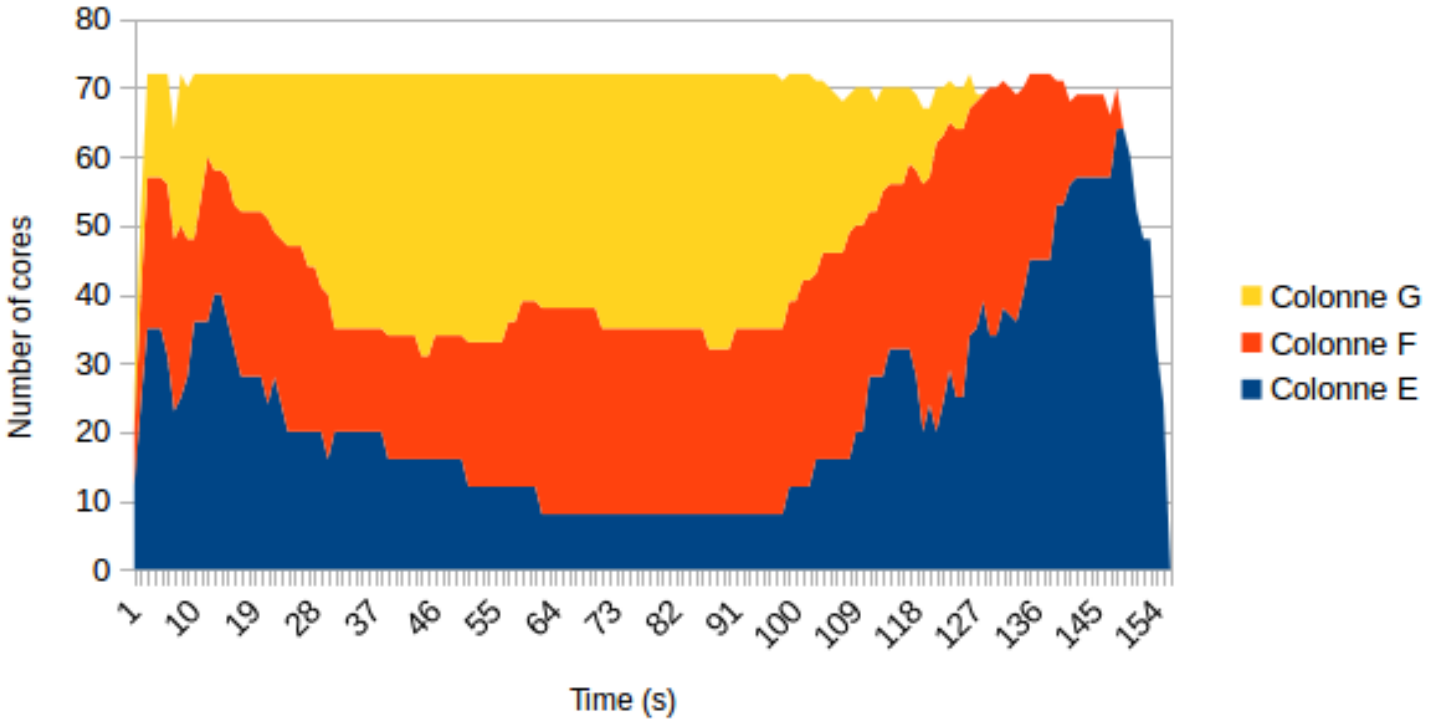
- Infrastructure :
  - 3 machines avec 16 cœurs
  - 1 machine avec 24 cœurs
- Services :
  - Premium : Priorité dans les 2 classe SLA $= 3$
  - Advanced : Priorité dans les 2 classe SLA $= 2$
  - Best effort : Priorité dans les 2 classe SLA $= 1$

Soumission des 360 requêtes en même temps



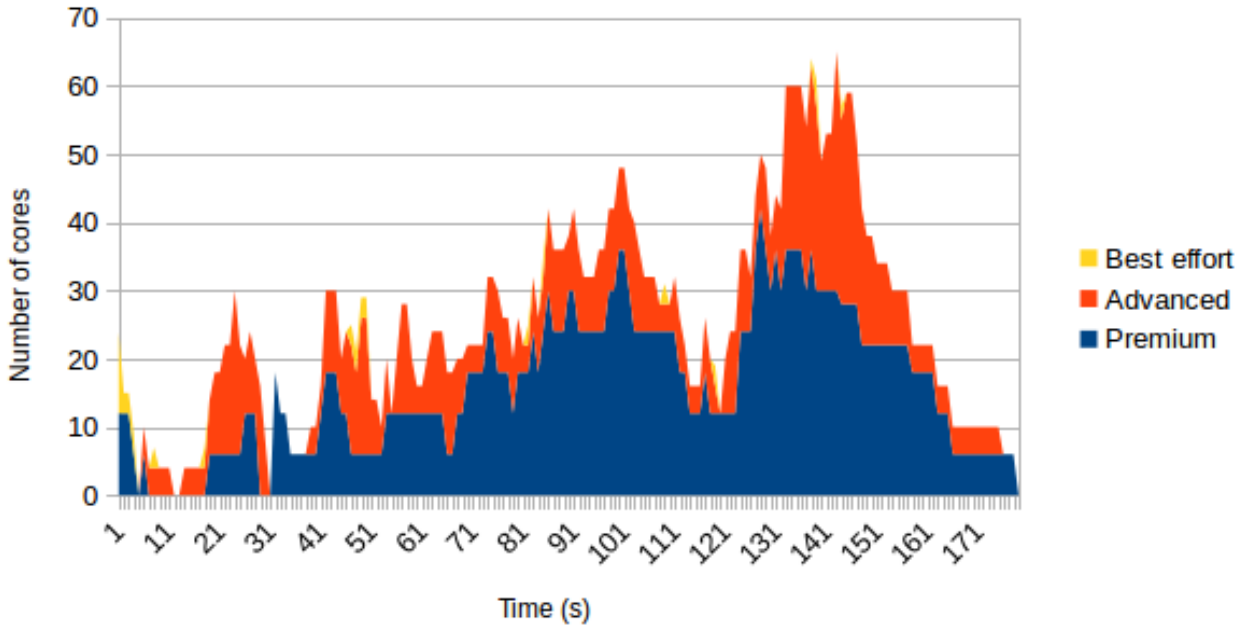
- 120 requêtes pour chaque service
- Le même service est fixé pour les deux classes SLAs

Soumission des 360 requêtes avec une fréquence fixe



- 120 requêtes pour chaque service
- Chaque 0.5 seconds 3 requêtes sont soumises (Premium, Advanced et Best effort)
- Le même service est fixé pour les deux classes SLAs

Soumission de 500 requêtes https ://prezi.com/scale/



- Trace de Prezi
- 52 requêtes de type Premium, 63 de type Advanced et 385 de type Best effort
- Le même service est fixé pour les deux classes SLAs

Outline

Conclusion & Perspectives

Conclusion

- Système d'allocation de ressources combiné avec un modèle économique
- Modèle économique avec deux classes SLA (qualitatif et quantitatif) et trois services pour chaque classe (Premium, Advanced et Best effort)
- Sélection en premier la requête qui a le bon compromis entre les deux classes SLAs
- Calcul du nombre de cœurs alloué pour chaque requête selon le modèle économique du client et la charge des machines
- Minimise le nombre des machines utilisées dans l'infrastructure
- Validation de notre système par des simulations

Conclusion & Perspectives

Perspectives

- Implémenter les heuristiques de décision dans un « vrai » système
- Faire des tests sur la platforme Grid5000, à large échelle
- Valider notre système avec d'autres traces réelles
- Introduire des heuristiques concientes des données

Merci pour votre attention