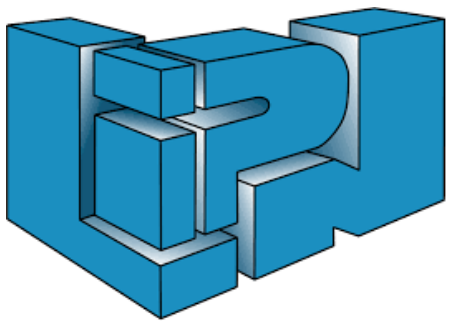
SAFC : Scheduling and Allocation Framework for Containers in a Cloud Environment

Tarek Menouer

PostDoc au laboratoire LIPN
Université Paris 13

03 Juillet 2018



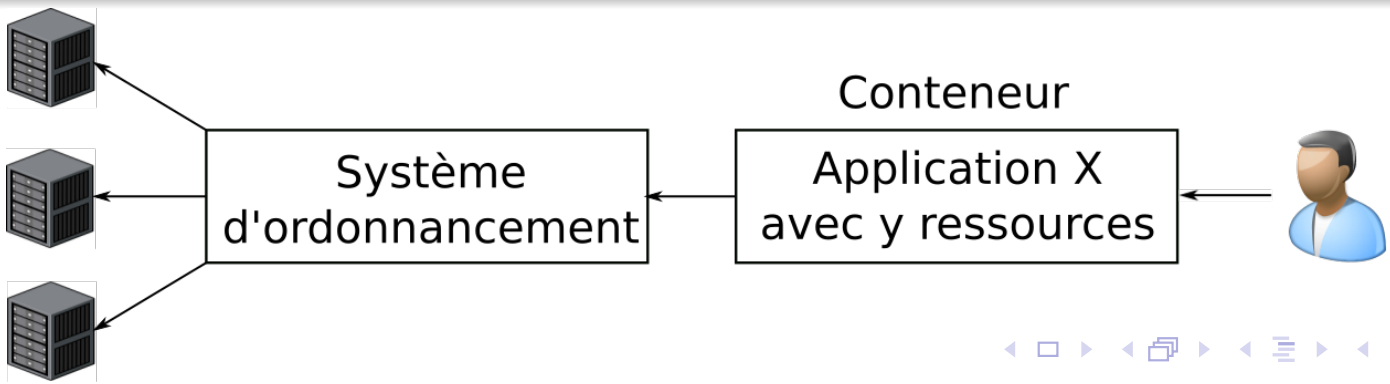
Outline

Outline

Contexte (1/2)

Contexte

- Conteneur est une enveloppe virtuelle qui permet de packager une application
  - Exécution d'une application parallèle
  - Le nombre de ressources (CPUs) est ajusté suivant le besoin de l'utilisateur
- Technologie des conteneurs connait un rythme d'adoption exponentiel poussé par le succès de Docker
- Systèmes d'ordonnancement de conteneurs dans le cloud existent :
  - Google Kubernetes, Docker Swarm, Apache Mesos, ...



Contexte (2/2)

Problème avec les systèmes d'ordonnancement traditionnels

- Destinés aux utilisateurs (clients) connaissaient le nombre de ressources qu'il faut pour exécuter une application en parallèle dans un conteneur
  - ✗ Le nombre de ressources allouées pour chaque conteneur est fixé statiquement par le client

Solution

- Un nouveau framework d'ordonnancement de conteneurs (SAFC: Scheduling and Allocation Framework for Containers)
  - ✓ Calcul dynamique du nombre de ressources allouées pour chaque conteneur
  - ✓ Modèle économique basé sur des classes SLA

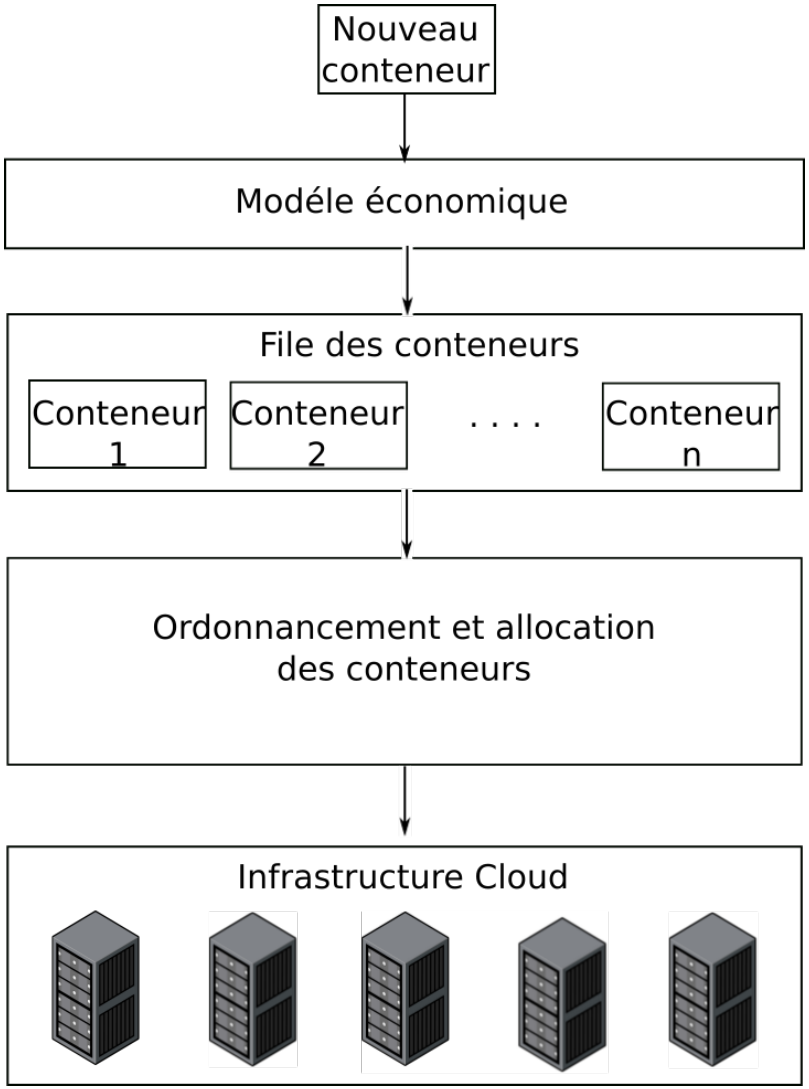
Outline

Scheduling and Allocation Framework for Containers

Objectif

- Répondre aux besoins d'une entreprise qui a une infrastructure privée et qui veut optimiser l'ordonnancement de plusieurs conteneurs soumis en ligne
  - Modèle économique afin de calculer le cout d'exécution et le nombre de ressources pour chaque conteneur
  - Maximiser le nombre de conteneurs exécutés
  - Minimiser la consommation énergétique

Architecture



Scheduling and Allocation Framework for Containers

Modèle économique

Modèle économique

- 2 Classes SLA qualitatives et 2 classes SLA quantitatives
- 3 services pour chaque classe :
  - Premium : Service le plus haut
  - Advanced : Service moyen
  - Best effort : Service le plus faible

Classes qualitatives et quantitatives

- Classes qualitatives :
  - Temps de satisfaction : temps d'attente du client
  - Réputation : Réputation des nœuds
- Classes quantitatives :
  - Nombre de ressources : Nombre de cœurs de calcul
  - Réplication: Réplication d'exécution des conteneurs pour répondre au problème de tolérance aux fautes

Principe général

Étapes de traitement

- Étape 1 : Ordonnancement des conteneurs
  - Choisir le premier conteneur c_i qui peut être exécuté

Scheduling and Allocation Framework for Containers

Principe général

Étapes de traitement

- Étape 1 : Ordonnancement des conteneurs
  - Choisir le premier conteneur c_i qui peut être exécuté
- Étape 2 : Sélection des nœuds candidats
  - Sélectionner pour le conteneur c_i un ensemble de nœuds candidats suivant le service de client au niveau de la classe réputation

Scheduling and Allocation Framework for Containers

Principe général

Étapes de traitement

- Étape 1 : Ordonnancement des conteneurs
  - Choisir le premier conteneur c_i qui peut être exécuté
- Étape 2 : Sélection des nœuds candidats
  - Sélectionner pour le conteneur c_i un ensemble de nœuds candidats suivant le service de client au niveau de la classe réputation
- Étape 3 : Calcul de ressources
  - Décider dynamiquement le nombre de cœurs affectés pour exécuter le conteneur c_i

Scheduling and Allocation Framework for Containers

Principe général

Étapes de traitement

- Étape 1 : Ordonnancement des conteneurs
  - Choisir le premier conteneur c_i qui peut être exécuté
- Étape 2 : Sélection des nœuds candidats
  - Sélectionner pour le conteneur c_i un ensemble de nœuds candidats suivant le service de client au niveau de la classe réputation
- Étape 3 : Calcul de ressources
  - Décider dynamiquement le nombre de cœurs affectés pour exécuter le conteneur c_i
- Étape 4 : Affectation de conteneurs
  - Calculer le nombre de réplications de c_i
  - Choisit les nœuds qui exécutent le conteneur c_x et ses réplications

Étape 1 : Ordonnancement des conteneurs (1/3)

Principe

- Chaque nouveau conteneur est sauvegardé dans une file globale
- Trier les conteneurs selon l'algorithme PROMETHEE II (Preference Ranking Organization METHod for Enrichment Evaluations)
  - Algorithme d'aide à la décision multi-critères
  - Combine des critères qualitatives et quantitatives

Scheduling and Allocation Framework for Containers

Étape 1 : Ordonnancement des conteneurs (2/3)

Algorithme PROMETHEE II

- Calculer pour chaque pair de conteneur (c_a et c_b) une distance (d_i) et un degré de préférence :

$$P_i(d_i) = \begin{cases} 0 & d_i \leq 0 \\ 1 & d_i > 0 \end{cases}$$

- Calculer pour chaque paire de conteneur (c_a et c_b) un index de préférence global : $\pi = \sum_{j \in C} p_j$
- Calculer pour chaque conteneur (c_a) un flux positif : $\phi^+(c_a) = \frac{1}{n-1} \sum_{x \in A} \pi(c_a, x))$
- Calculer pour chaque conteneur (c_a) un flux négatif : $\phi^-(c_a) = \frac{1}{n-1} \sum_{x \in A} \pi(x, c_a)$
- Calculer pour chaque conteneur (c_a) un flux global : $\phi(c_a) = \phi^+(c_a) - \phi^-(c_a)$
- Classer les conteneurs selon leur valeur de flux global

Scheduling and Allocation Framework for Containers

Étape 1 : Ordonnancement des conteneurs (3/3)

Conteneurs	Qualitatives		Quantitatives	
c_a	1	1	2	1
c_b	2	2	1	2
c_c	3	3	3	3

- 3 : Service Premium
- 2 : Service Advanced
- 1 : Service Best effort

Étape 1 : Ordonnancement des conteneurs (3/3)

Conteneurs	Qualitatives		Quantitatives	
c_a	1	1	2	1
c_b	2	2	1	2
c_c	3	3	3	3

- 3 : Service Premium
- 2 : Service Advanced
- 1 : Service Best effort

Paire de conteneurs	Distance				Degré de préférence				index de préférence global
	Qualitatives		Quantitatives		Qualitatives		Quantitatives		
(c_a, c_b)	-1	-1	1	-1	0	0	1	0	1
(c_a, c_c)	-2	-2	-1	-2	0	0	0	0	0
(c_b, c_a)	1	1	-1	1	1	1	0	1	3
(c_b, c_c)	-1	-1	-2	-1	0	0	0	0	0
(c_c, c_a)	2	2	1	2	1	1	1	1	4
(c_c, c_b)	1	1	2	1	1	1	1	1	4

Étape 1 : Ordonnancement des conteneurs (3/3)

Conteneurs	Qualitatives		Quantitatives	
c_a	1	1	2	1
c_b	2	2	1	2
c_c	3	3	3	3

- 3 : Service Premium
- 2 : Service Advanced
- 1 : Service Best effort

Paire de conteneurs	Distance				Degré de préférence				index de préférence global
	Qualitatives		Quantitatives		Qualitatives		Quantitatives		
(c_a, c_b)	-1	-1	1	-1	0	0	1	0	1
(c_a, c_c)	-2	-2	-1	-2	0	0	0	0	0
(c_b, c_a)	1	1	-1	1	1	1	0	1	3
(c_b, c_c)	-1	-1	-2	-1	0	0	0	0	0
(c_c, c_a)	2	2	1	2	1	1	1	1	4
(c_c, c_b)	1	1	2	1	1	1	1	1	4

Conteneurs	ϕ^+	ϕ^-	ϕ	Classement
c_a	1	7	-6	**3**
c_b	3	5	-2	**2**
c_c	8	0	8	**1**

Étape 2 : Sélection des nœuds candidats

Principe

- Partitionnement statique de l'ensemble de nœuds en 3 classes :
  - Nœuds avec une haute réputation ⇒ Service Premium
  - Nœuds avec une réputation moyenne ⇒ Service Advanced
  - Nœuds avec une faible réputation ⇒ Service Best effort

Dans le futur

- Partager les nœuds en :
  - Nœuds privés (mini cloud privé)
  - Nœuds publics (min cloud public)
  - Nœuds hybrides (mini cloud hybrid)
- Appliquer des techniques de clustering permettant la décomposition des nœuds (selon la fiabilité des disques)

Scheduling and Allocation Framework for Containers

Étape 3 : Calcul de ressources (1/4)

Bornes Hard et Soft

- Fixer des bornes Hard et Soft pour chaque service
  - N'exécute pas un conteneur avec un faible service en utilisant plus de cœurs de calcul qu'un conteneur avec un haut service
- Bornes Hard fixées suivant la configuration du nœud le plus petit dans l'infrastructure
- Bornes Soft fixées suivant la charge des nœuds candidats

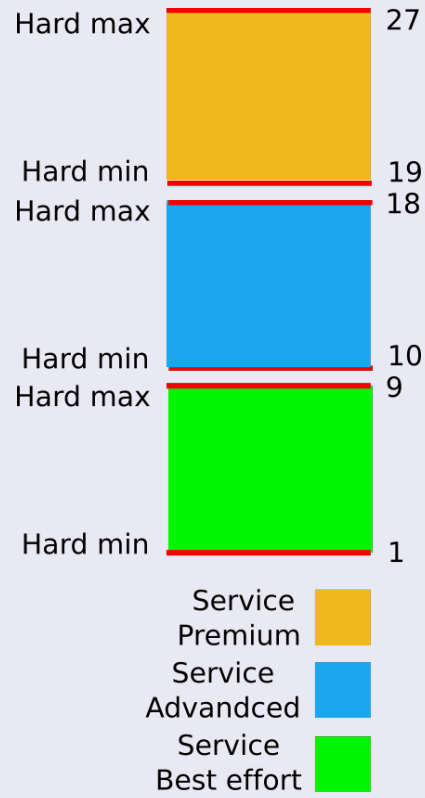
Scheduling and Allocation Framework for Containers

Étape 3 : Calcul de ressources (2/4)

Bornes Hard

- K: le nombre de cœurs du nœud le plus petit
- Chaque service dans la classe nombre de ressources a un Hard min et Hard max cœurs :
  - Service Best effort : Hard min $= 1$ et Hard max $= \frac{K}{3}$;
  - Service Advanced : Hard min $=$ Hard max de service Best effort service $+1$ et Hard max $= 2 \times \frac{K}{3}$;
  - Service Premium : Hard min $=$ Hard max de service Advanced $+1$ et Hard max$= K$

Exemple



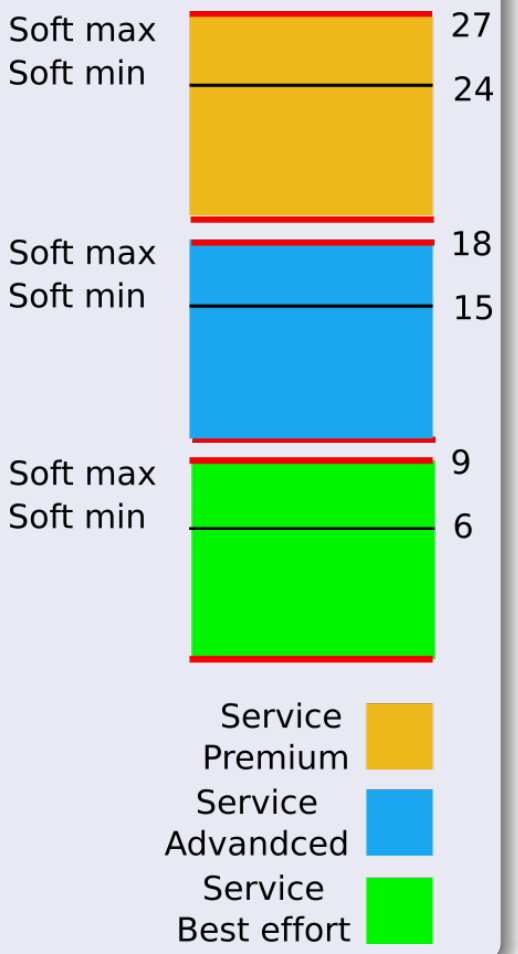
Scheduling and Allocation Framework for Containers

Étape 3 : Calcul de ressources (3/4)

Bornes Soft

- D = Hard Max cœurs - Hard Min cœurs+1
- Si la charge des nœuds entre 0% et 33% :
  - Soft Min cœurs = Hard Min cœurs + $2 \times \frac{D}{3}$ et Soft Max cœurs = Hard Max cœurs

Exemple

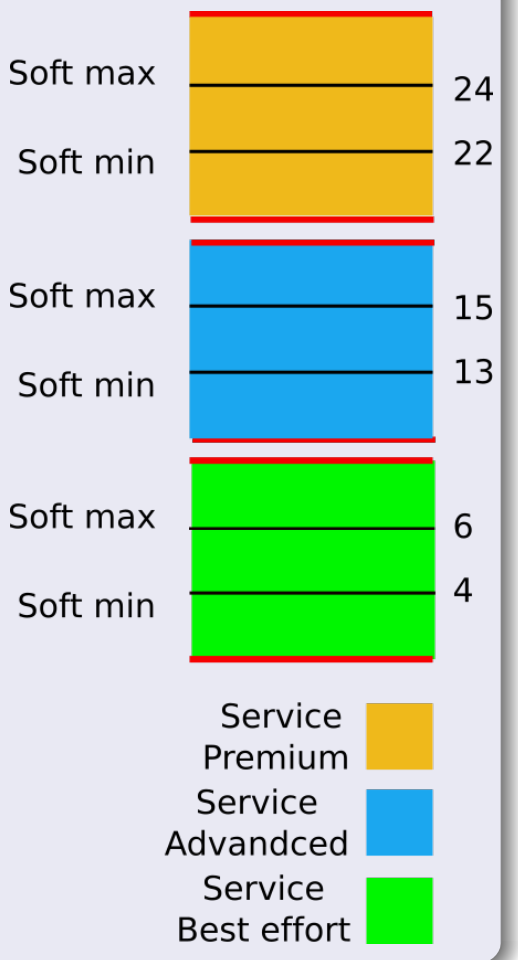


Étape 3 : Calcul de ressources (3/4)

Bornes Soft

- D = Hard Max cœurs - Hard Min cœurs+1
- Si la charge des nœuds entre 0% et 33% :
  - Soft Min cœurs = Hard Min cœurs + $2 \times \frac{D}{3}$ et Soft Max cœurs = Hard Max cœurs
- Si la charge des nœuds entre 34% et 66% :
  - Soft Min cœurs = Hard Min cœurs + $\frac{D}{3}$ et Soft Max cœurs = Hard Min cœurs + $2 \times \frac{D}{3}$ -1

Exemple

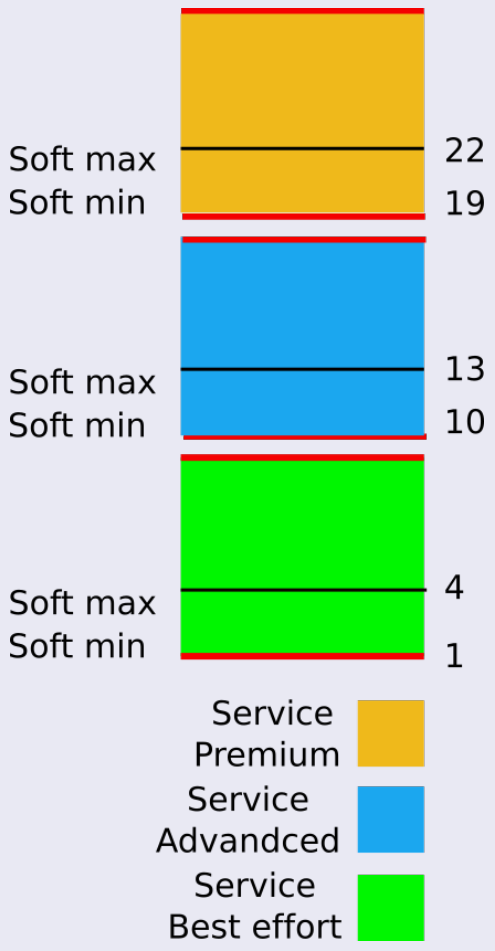


Étape 3 : Calcul de ressources (3/4)

Bornes Soft

- D = Hard Max cœurs - Hard Min cœurs+1
- Si la charge des nœuds entre 0% et 33% :
  - Soft Min cœurs = Hard Min cœurs + $2 \times \frac{D}{3}$ et Soft Max cœurs = Hard Max cœurs
- Si la charge des nœuds entre 34% et 66% :
  - Soft Min cœurs = Hard Min cœurs + $\frac{D}{3}$ et Soft Max cœurs = Hard Min cœurs + $2 \times \frac{D}{3}$ -1
- Si la charge est supérieure à 66% :
  - Soft Min cœurs = Hard Min cœurs et Soft Max cœurs = Hard Min cœurs + $\frac{D}{3}$ -1

Exemple



Scheduling and Allocation Framework for Containers

Étape 3 : Calcul de ressources (4/4)

Calcul de nombre de cœurs

- x_i= nombre de cœurs calculé pour le conteneur c_i
- C=$\{c_1, \cdots, c_n\}$ (ensemble des conteneurs sauvegardé dans la file)
- P= $\{p_1, \cdots, p_n\}$ (priorités dans la classe nombre de ressources)
- T = nombre de cœurs disponibles dans tous les nœuds candidats

$$\pi = \begin{cases} 1 & if \quad (T - \sum_{j=0}^{n} Soft\ Min_j) > 0 \\ 0 & else \end{cases}$$

- $x_i = Soft\ min_i + \left\lfloor \frac{\pi \times p_i \times (T - \sum_{j=0}^{n} Soft\ min_j)}{\sum_{j=0}^{n} p_j} \right\rfloor$
- If $x_i >$ Soft Max cœurs, $x_i =$ Soft Max cœurs
- If $x_i <$ Soft Min cœurs, $x_i =$ Soft Min cœurs

Étape 4 : Affectation de conteneurs (1/2)

Calcul des réplications

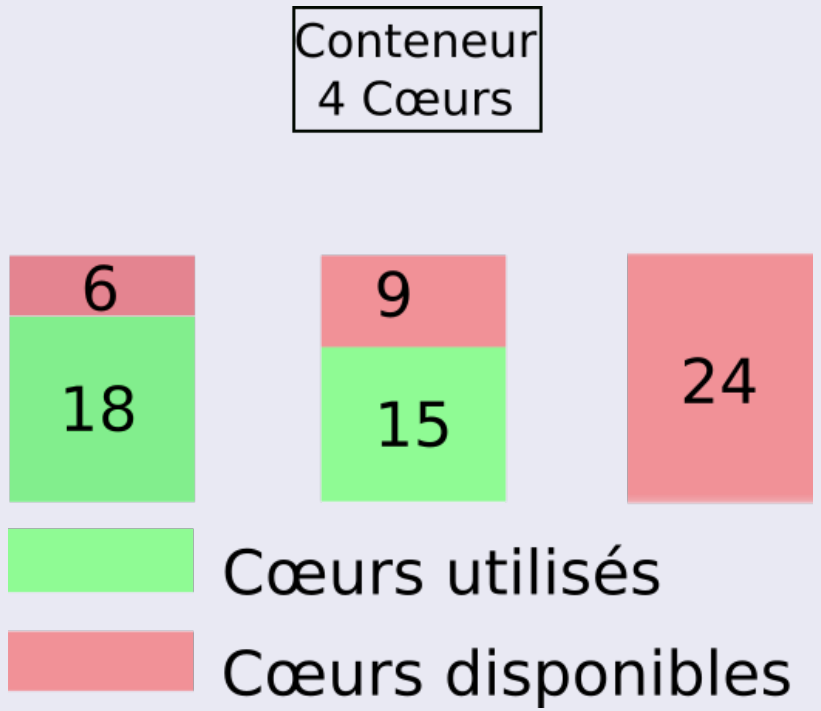
- Calculer le nombre de réplications suivant le service de client :
  - Service Premium $\Rightarrow$ 3 réplications
  - Service Advanced $\Rightarrow$ 2 réplications
  - Service Best effort $\Rightarrow$ 1 réplication

Choix des nœuds

- Affecter le conteneur et ses réplications au nœuds suivant la politique de Bin packing
  - Affecter le conteneur c_i au nœud n_j qui a le plus petit nombre de ressources disponibles
  - Réduire le nombre de nœuds actifs $\Rightarrow$ minimiser la consommation énergétique

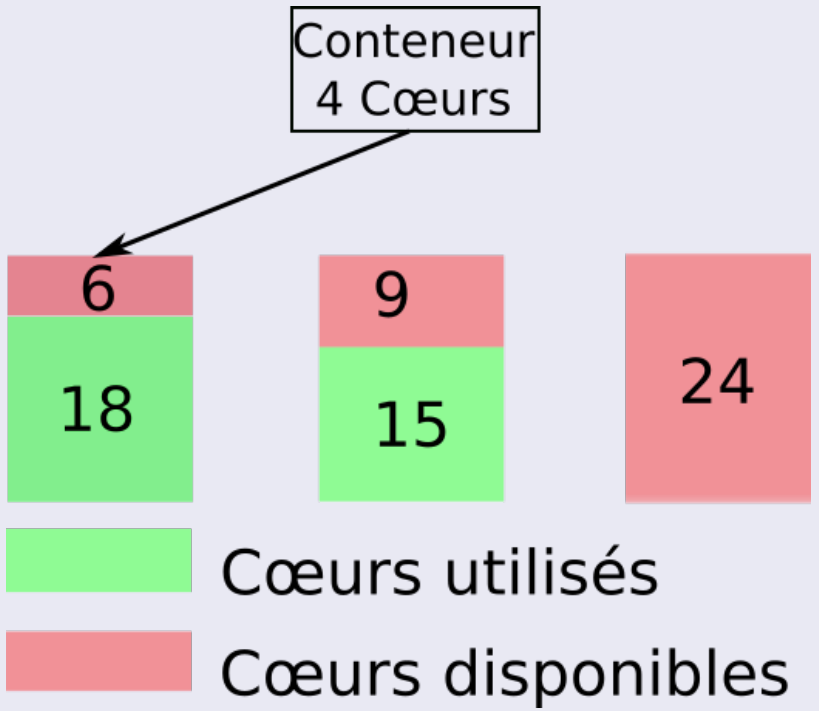
Étape 4 : Affectation de conteneurs (2/2)

Stratégie BinPacking



Étape 4 : Affectation de conteneurs (2/2)

Stratégie BinPacking



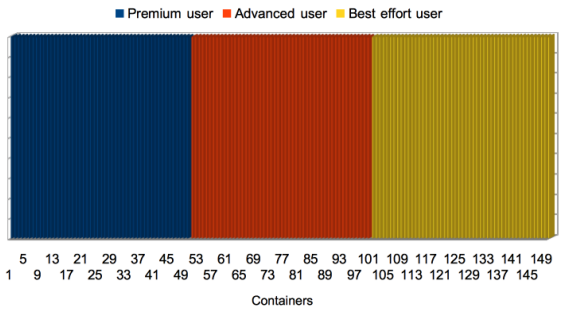
Outline

Protocole

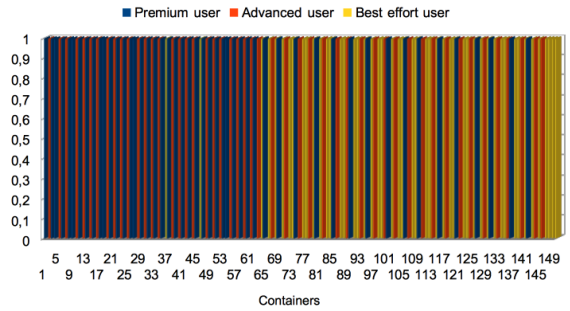
Protocole d'exécution

- Infrastructure : 32 nœuds utilisant en total 1024 cœurs de calcul
- Tous les conteneurs sont soumis par 3 clients :
  - Client 1: Service Premium dans les 4 classes SLA
  - Client 2: Service Advanced dans les 4 classes SLA
  - Client 3: Service Best effort dans les 4 classes SLA
- Temps d'exécution de chaque conteneur (c_i) égale à $\frac{30 minutes}{P}$ (P = nombre de cœurs calculé pour c_i)

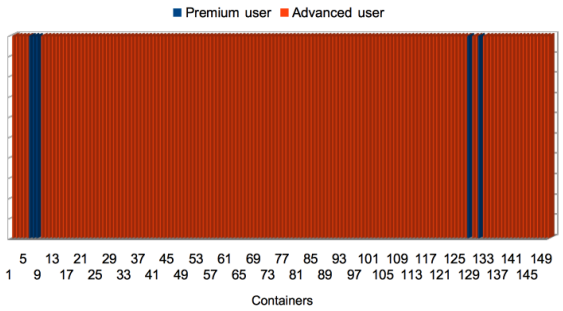
Performance de notre framework



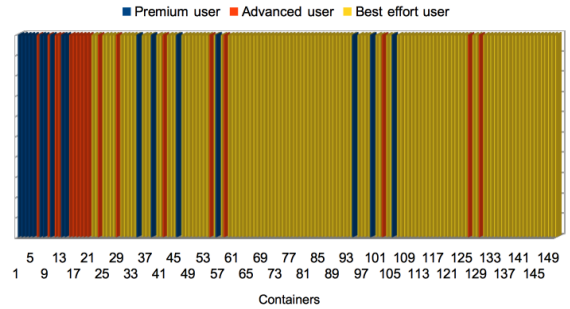
Soumission de 150 conteneurs LXC en même temps



Soumission de 150 conteneurs LXC avec une fréquence fixe



Soumission de 150 conteneurs LXC suivant les traces de Google



Soumission de 150 conteneurs LXC suivant les traces Prezi

Comparaison de temps d'exécution

Type de soumission	Calcul de nombre de cœurs	Temps de calcul	Nombre de cœurs moyen pour chaque conteneur		
			client 1	client 2	client 3
Soumission en même temps	Min cores	3810.09	22	11	1
	Max cores	1990.05	32	21	10
	Dynamic computation	2075.06	28	18	10
Soumission avec une fréquence fixe	Min cores	4730	22	11	1
	Max cores	2040.05	32	21	10
	Dynamic computation	2100.06	30	19	9
Soumission suivant les traces Google	Min cores	2315.05	22	11	-
	Max cores	1850.04	32	21	-
	Dynamic computation	1875.04	29	18	-
soumission suivant les traces Prezi	Min cores	4815.13	22	11	1
	Max cores	2270.06	32	21	10
	Dynamic computation	2380.06	29	17	9

- Notre méthode de calcul dynamique est $\approx 3\%$ plus lente que la méthode max cœurs
- Notre méthode de calcul dynamique consomme $\approx 10\%$ moins de cœurs que la méthode max cœurs

Outline

Conclusion et Perspectives

Conclusions

- Un nouveau framework d'ordonnancement et d'allocation de conteneurs
- Modèle économique avec 2 classes SLA qualitatives et 2 classes SLA quantitatives
- 3 services pour chaque classe SLA
- Calcul automatique du nombre de cœurs alloués pour chaque conteneur

Perspectives

- Utiliser un algorithme d'apprentissage pour adapter les bornes soft suivant la fréquence de soumission des conteneurs
- Proposer un mécanisme de consolidation pour adapter le nombre de nœuds actifs afin de réduire la consommation énergétique

Merci pour votre attention

Plus d'information sont disponibles sur :

`https://lipn.univ-paris13.fr/~menouer/Wolphin.html`
`https://lipn.univ-paris13.fr/~menouer/`

Travaille réaliser dans le cadre de

projet " *FUI-22 Wolphin*"