Collecte de ressources libres dans une grille en préservant le système de fichiers: une approche autonomique

COMPAS 2021, Lyon

Quentin GUILLOTEAU,[*] Olivier RICHARD,[*] Bogdan ROBU,[**]
Éric RUTTEN[*]

[*]Université Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG

[**]Université Grenoble Alpes, CNRS, Grenoble INP, GIPSA-lab

2021-07-09

Contexte

HPC

Systèmes HPC de + en + complexes:

- imprévisible en performances
- imprévisible en consommation d'énergie
- imprévisible en temps d'accès mémoire.

Problèmes Potentiels

- Surcharge
- Surchauffe

↪ nécessité d'**une régulation en ligne** pour assurer les performances

Une solution possible

Informatique Autonomique

Informatique Autonomique

Définition (IBM 2000s)

Systèmes pouvant **s'auto-réguler** étant donnés des **objectifs de haut-niveau** par les administrateurs

Outil Principal: La Boucle MAPE-K

Actionneur(s) et Capteur(s)
(Ex: réguler T° CPU avec sa freq)

1. **M**onitor (Capte la T°)
2. **A**nalyse (Erreur signifiante ?)
3. **P**lan (Quelle freq choisir ?)
4. **E**xecute (Applique nouv. freq.)
5. **K**nowledge (Freq. min/max)

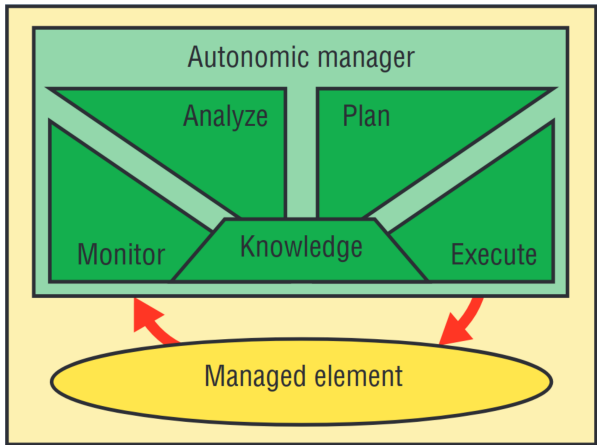


Figure: La Boucle MAPE-K

Théorie du Contrôle

Théorie du Contrôle

Régule le comportement de systèmes dynamiques

↪ Interprétation de la Boucle MAPE-K

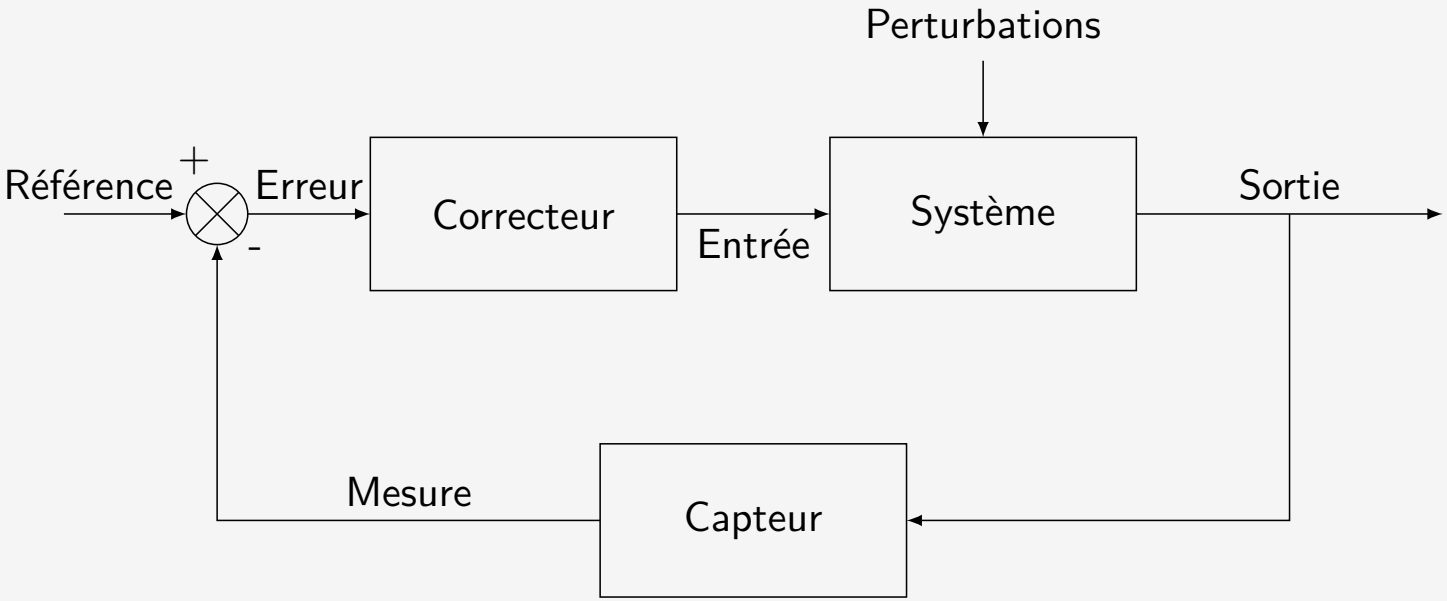


Figure: Boucle de contrôle

Collecte de ressources libres

Ordonnancement laisse des "trous"

- spécifications
- mauvaise estimation du temps d'exécution
- ...

Objectif

Utiliser les ressources libres pour exécuter des tâches moins prioritaires

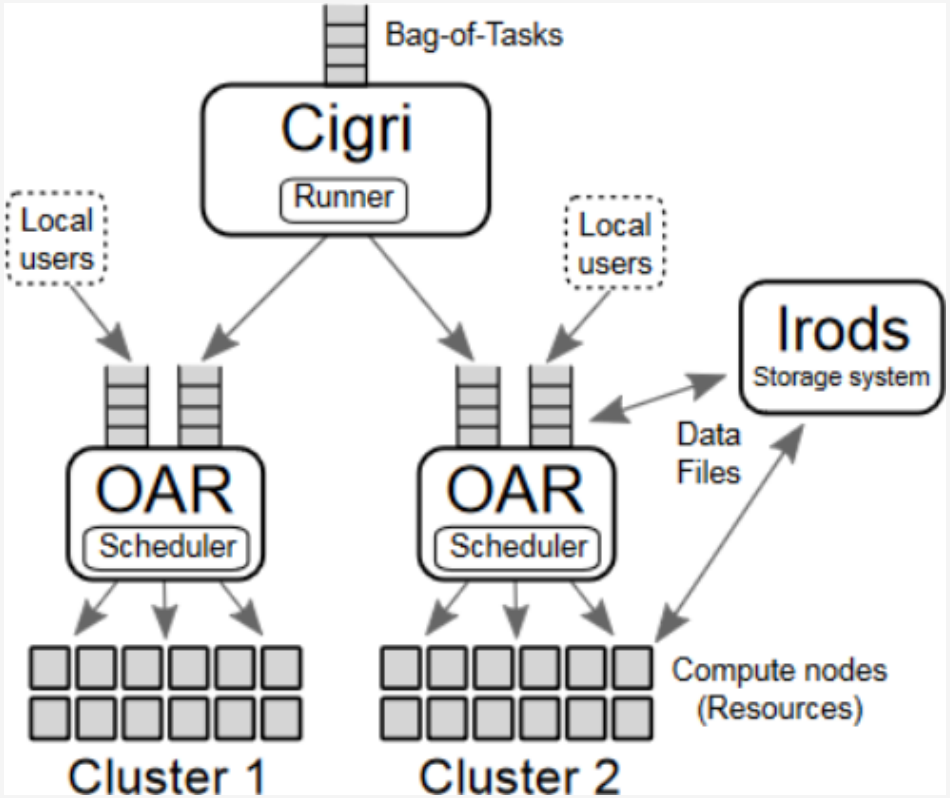
État de l'art

- BOINC/Condor: récupère des cycles CPU de machines inactives
- OurGrid: partage de machines multi-labos
- BeBiDa: tâches BigData sur machines libres HPC

CiGri: Présentation

CiGri (CIMENT Grid)

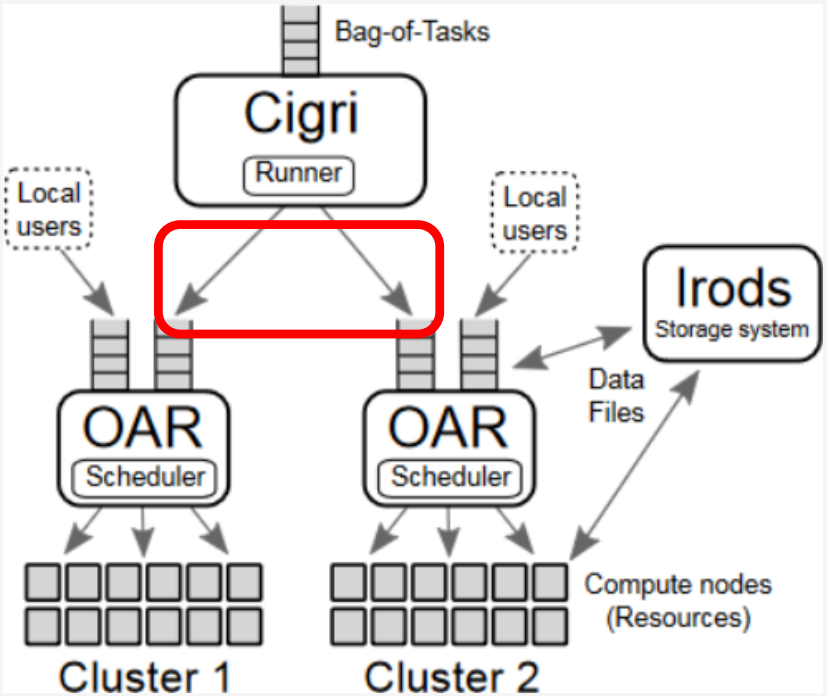
- Intergiciel résistant aux fautes pour grille légère
- Interagit avec un ensemble d'ordonnanceurs *OAR*
- **But**: Exploiter les ressources libre d'une grille de manière **non intrusive**
- **bag-of-tasks**: Large ensemble de tâches multi-paramétrique
- **Tâches Best-effort**: Tâches avec priorité la plus basse



CiGri: Boucle de Soumission (1/2)

Algorithm 1: Boucle de Soumission

```
rate = 3;
increase_factor = 1.5;
while tâches non exécutées do
    if pas de tâches en cours
     d'exécution then
        soumet rate tâches;
        rate = min(rate ×
         increase_factor, 100);
    end
    while nombre de tâches en cours
     d'exécution > 0 do
        dort pendant 30 secondes;
    end
end
```

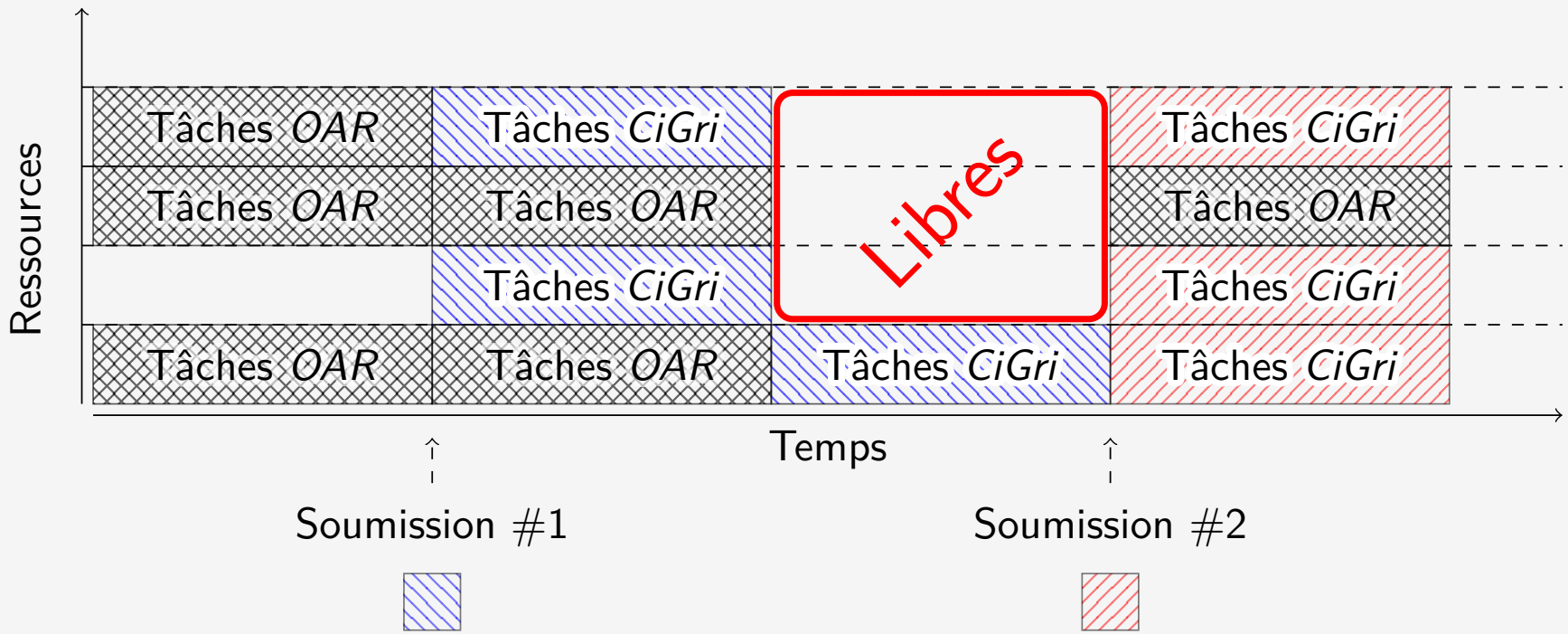


CiGri: Boucle de Soumission (2/2)

Le Problème

Doit attendre la fin de la soumission précédente pour soumettre à nouveau

↪ réduit la surcharge du système mais peut introduire une **sous-utilisation** des ressources



CiGri: Le besoin d'amélioration

Observation

Algo actuel de *CiGri*: **trop protecteur**
↪ peut être amélioré si **prise en compte du l'état de la grille**

L'idée générale

Appliquer l'informatique autonomique à *CiGri*

Réguler le nombre de tâches soumisses à *OAR* en fonction du nombre de ressources libres de la grille
+ ajout de contraintes (e.g. charge du système de fichiers distribué)

Système de Fichiers Distribué et son capteur

Système de Fichiers Distribué (SFD)

stocke les fichiers des utilisateurs de la grappe

Problème Potentiel

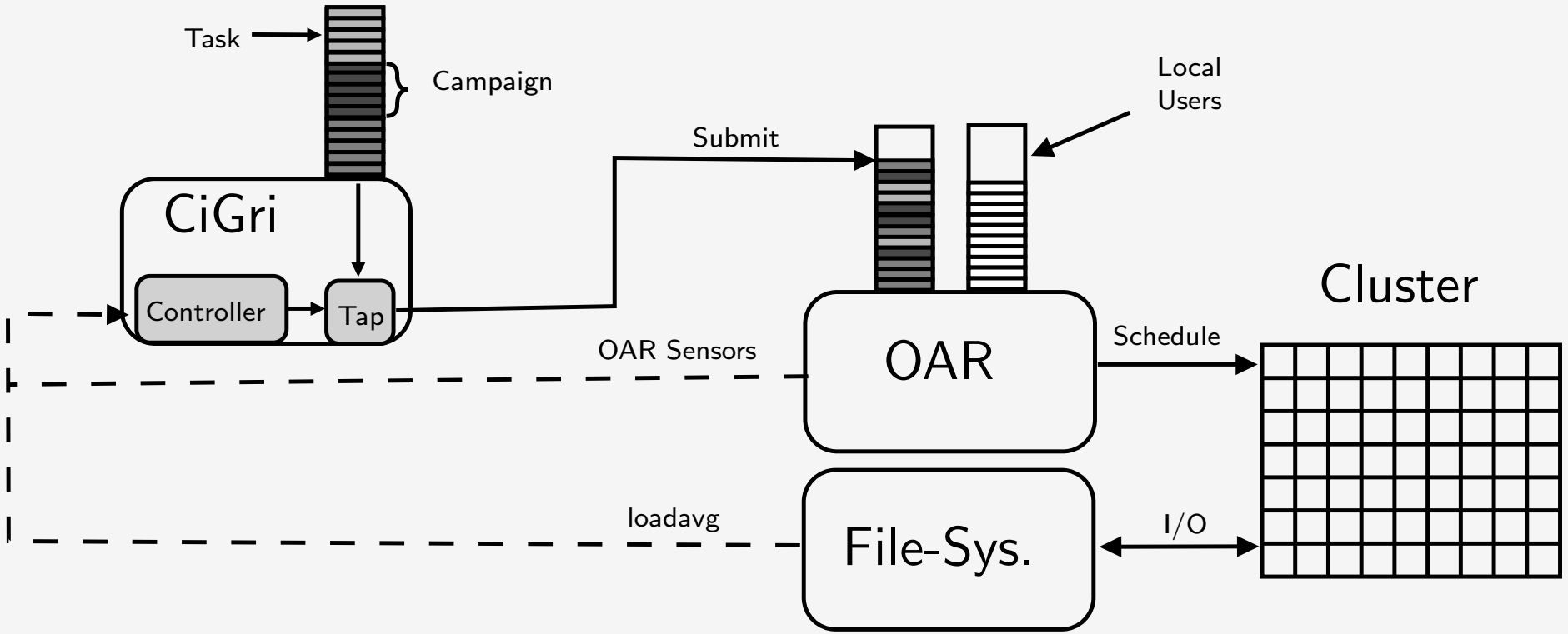
Trop de lectures/écritures simultanées $\Longrightarrow \nearrow$ temps lecture/écriture
$\hookrightarrow$ perturbation des tâches utilisateurs prioritaires

$\hookrightarrow$ Comment capter la surcharge du SFD ?

Notre capteur: `/proc/loadavg`

- Représente le nombre de processus CPU en cours d'exécution ou en attente pour le disque
- Lissage exponentiel $\rightarrow$ Inertie $\rightarrow$ cool pour contrôle

CiGri: ses boucles de rétro-action, capteurs et actionneurs



Définition du Problème et Travaux Précédents

Objectif

Utiliser le plus de ressources libres de la grappe **sans surcharger** le serveur de fichiers distribué

Travaux précédents

- Correcteur PI pour réguler la quantité le tâches soumisses à *OAR*:
  - meilleur utilisation des ressources vs. solution originale
  - ~~serveur fichiers distribué~~
- Correcteur MPC avec prise en compte du SFD:
  - Modèle du système (trop) simpliste
  - Ne passait pas à l'échelle

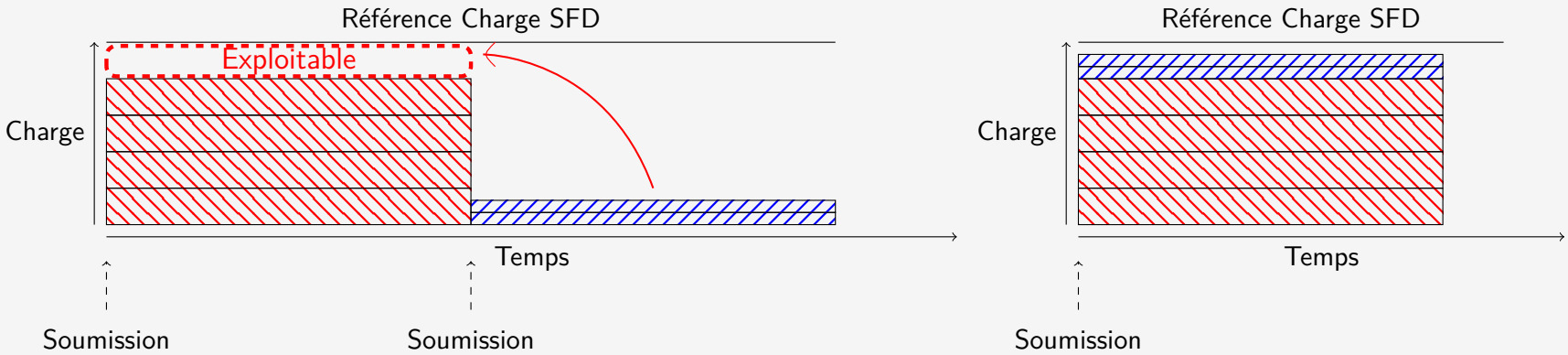
Observation

Problème

Originellement: soumissions de *CiGri* composées uniquement de tâches provenant de la **même campagne**

$\hookrightarrow \simeq$ même comportement (temps exec. + E/S)

$\hookrightarrow$ peut amener à une sous utilisation de la grappe



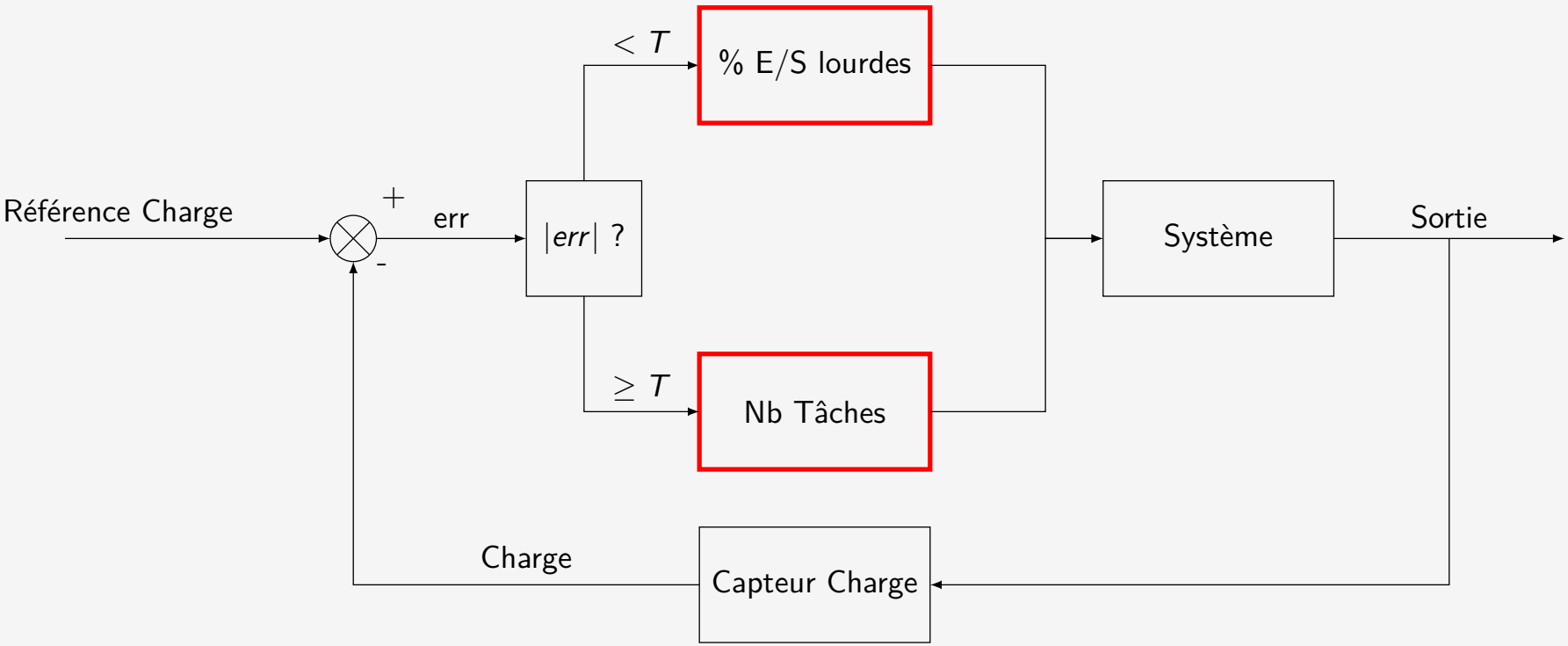
Stratégie

Soumettre des tâches de 2 campagnes avec des charges E/S $\neq$

Régulation Proposée: Approche Biphasique

Deux modes/phases de contrôle

1. Le **nombre total de tâches** soumisses à *OAR* ("big step")
2. Le **pourcentage de tâches E/S lourdes** soumisses ("small step")



Quelques détails supplémentaires

Ajout d'un garde fou

- Correcteur sur la file d'attente *OAR*
- But: éviter de "perdre la main"
- Nb tâches = min des sortie des correcteurs

Correcteur Intégral: (Δréponse $\propto$ erreur)

$$U_k = U_{k-1} + K \times Erreur_k$$

$$= U_{k-1} + K \times (Ref - Charge_k) = U_0 + K \sum_{i=0}^{k} Erreur_i$$

Dispositif Expérimental

Architecture

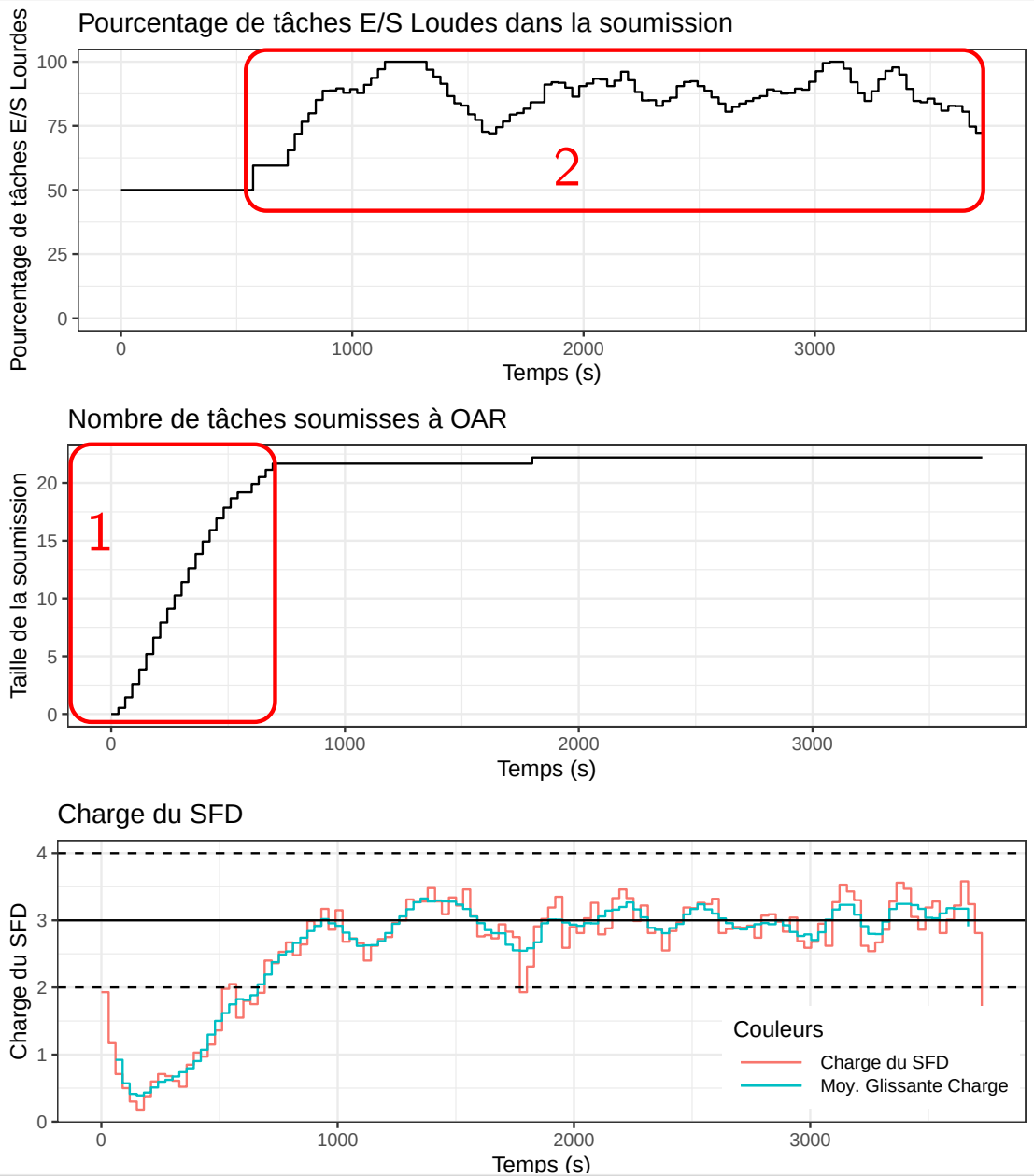
4 noeuds de Grid'5000 (Grisou)

- 1 Serveur *OAR* (v3)
- 1 Serveur *CiGri*
- 1 Serveur de fichiers distribué (NFS)
- émulation d'une grappe de 100 ressources

Expérience

- Campagne de 1000 tâches: `sleep` 30s + écrit 100Mo
- Campagne de 1000 tâches: `sleep` 30s + écrit 10Mo
- Référence de Charge = 3, Threshold = 1
- Utilisateurs Prioritaires: comportement synthétique

Résultats sans utilisateurs prioritaires



Remarques

1. Phase de montée ("big steps")
2. Puis "small steps"
3. Garde la charge dans l'intervalle

Résultats avec utilisateurs prioritaires

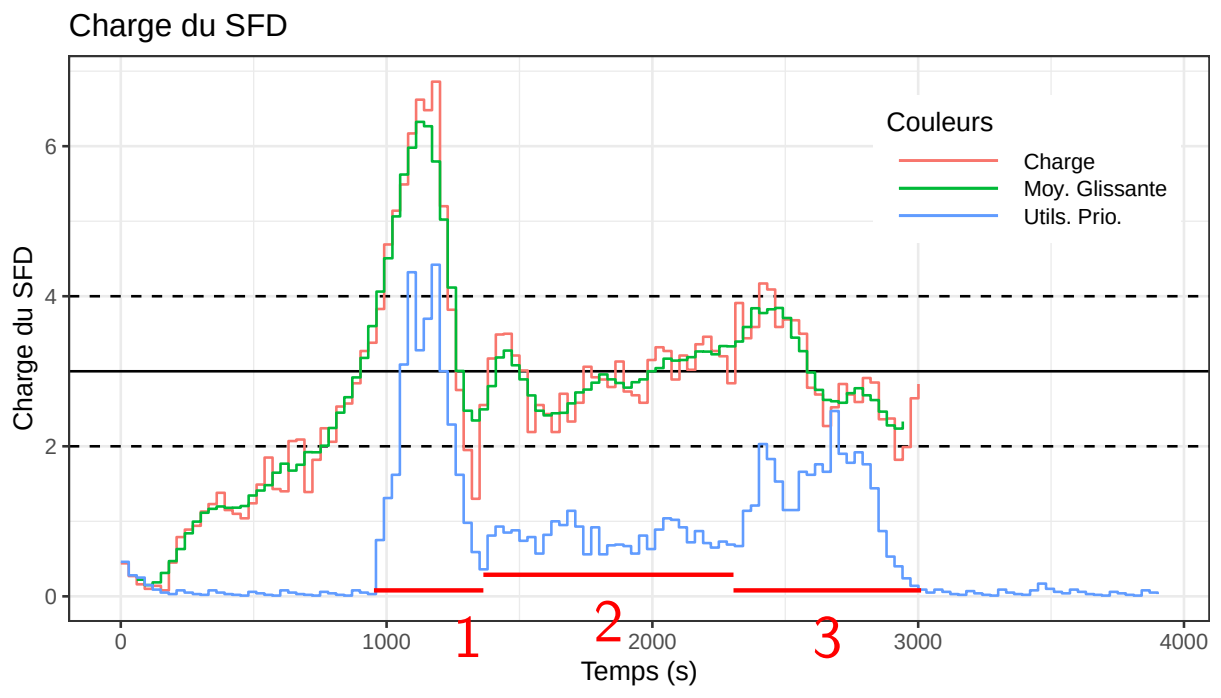


Figure: Charge du SFD avec Utilisateurs Prioritaires

Remarques

- Charge synthétique (3 Phases)
  - Réaction lente aux variations trop brusques
  + Sinon, parvient à rester dans l'intervalle

Comparaison

Rappel de l'objectif

Utiliser le + de ressources libres de la grappe **sans surcharger** le SFD

Stratégie	Rétro-action	Proportion du temps avec le SFD surchargé (%)			Util. grappe (%)	
		Charge > Ref	Charge > Ref + 33%	Charge > Ref + 66%	absence	présence
Originelle	✗	80.49%	75.61%	60.98%	63.7	73.8
Scan	✓	17.86%	7.14%	3.57%	13.8	16.1
MPC	✓	89.02%	84.15%	59.76%	28.2	44.5
Contrôle Simple	✓	25.93%	11.11%	7.40%	17.4	23.0
Contrôle Biphasé	✓	37%	10%	7%	19.6	25.8

Remarques

- Bonne régulation de la charge pour Biphasé (Top 3)
- Meilleure utilisation des ressources dans ce Top 3
- Meilleure utilisation // au contrôle simple → montre l'intérêt de l'approche biphasée

Conclusion & Perspectives

Conclusion

- Solution permettant de réguler la charge du SFD autour d'une valeur désirée
- tout en récoltant davantage de ressources

Perspectives

- Lien `loadavg` et impact temps lecture/écriture. Autre/combinaison capteurs
- Détection du type de campagne (E/S Lourde ou E/S Légère)
- Charges réelles/Benchmarks
- Identification des gains des correcteurs
- autres formes de régulation (PID, RST, etc)