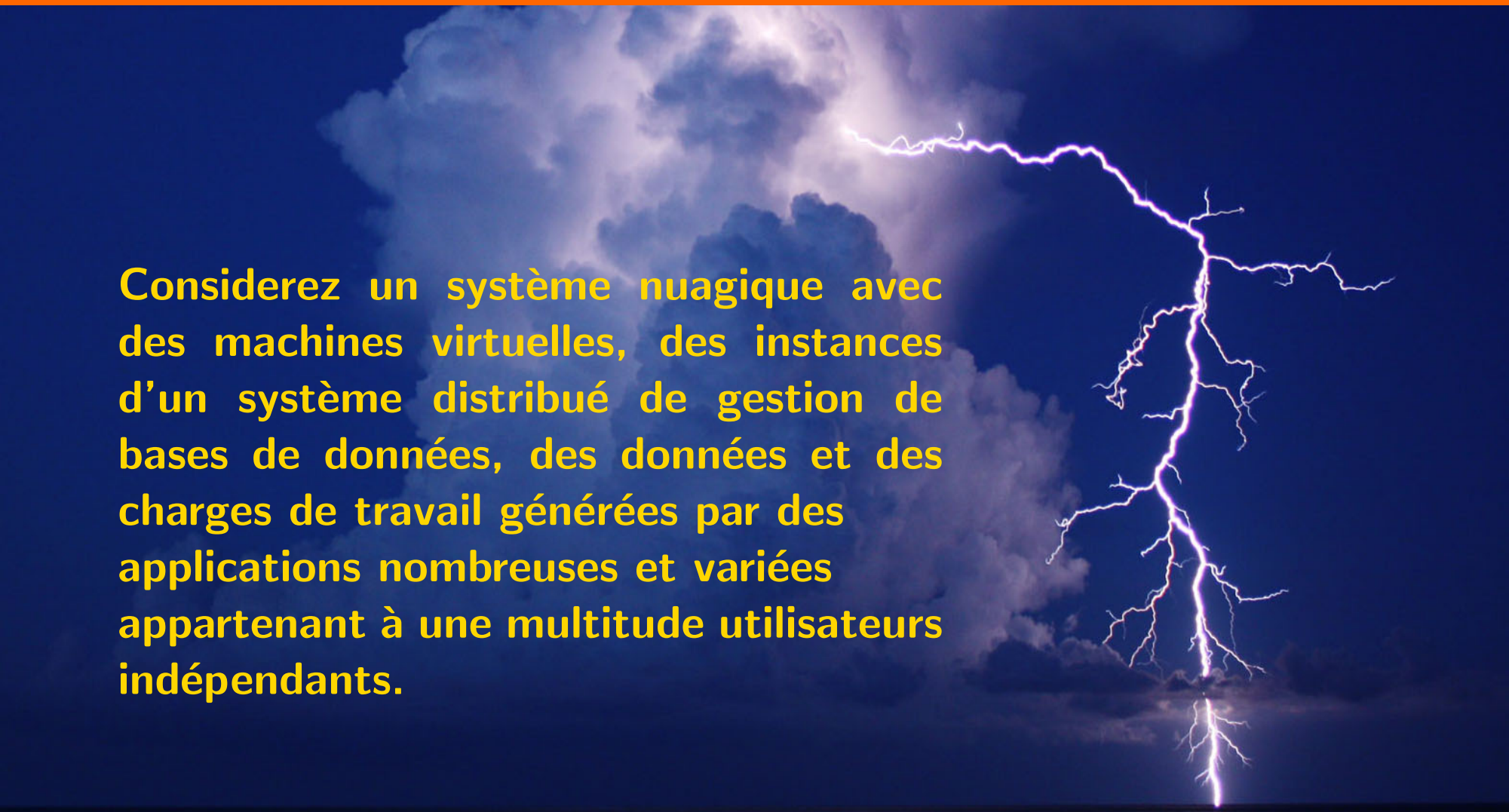
Apprentissage par Renforcement du Réglage des Configurations des Systèmes

Debabrota Basu[1], Qian Lin[1], Zihong Yuan[1], Pierre Senellart[1,2,3], Stéphane Bressan[1,3]

[1]Ecole d'Informatique, Université Nationale de Singapour, Singapour
[2]Institut Mines–Télécom; Télécom ParisTech; CNRS LTCI, France
[3]Image & Pervasive Access Lab, UMI CNRS 2955

Motivation



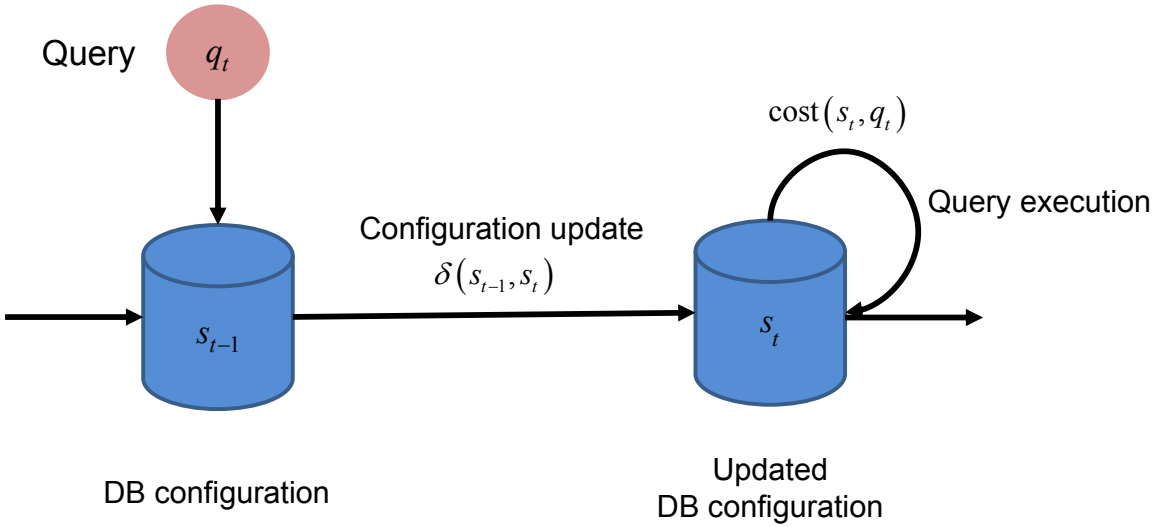
Considerez un système nuagique avec des machines virtuelles, des instances d'un système distribué de gestion de bases de données, des données et des charges de travail générées par des applications nombreuses et variées appartenant à une multitude utilisateurs indépendants.

Comment configurer un tel système?

Contribution

- Nous proposons une approche de **réglage des configurations** des systèmes sans modèle de coût;
- Nous modélisons le processus de réglage par un **processus de décision Markovien**;
- Nous utilisons l'**apprentissage par renforcement** pour apprendre le modèle de coût et optimiser les décisions;
- Nous presentons **COREIL**, un application de notre approche au **réglage des index** dans les systèmes de gestion de bases de données.

Réglage de Configuration



Per-stage cost $C(s_{t-1}, s_t, q_t) = \delta(s_{t-1}, s_t) + \text{cost}(s_t, q_t)$

Processus de Décision Markovien

- **Etat**: Configurations $s \in S$ et réception de la requête q_t;
- **Action**: Changement de configuration $s_{t-1} \rightarrow s_t$ et exécution de la requète q_t;
- **Fonction de coût par étape**: Coût du changement de configuration et coût d'exécution de la requête:

$$C(s_{t-1}, s_t, q_t) = \delta(s_{t-1}, s_t) + cost(s_t, q_t)$$

- **Politique**: Fonction de changements de configuration en fonction des configurations et des requètes.

Enoncé du Problème

- Pour une politique π et un facteur de dévaluation γ $(0 < \gamma < 1)$ la **fonction cumulative de coût** est définie par l'équation:

$$V^{\pi}(s) \triangleq \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} C(s_{t-1}, s_t, q_t)\right] \text{ telle que } \begin{cases} s_0 = s \\ s_t = \pi(s_{t-1}, q_t), \\ t \geq 1 \end{cases}$$

- **Objectif**: Trouver une politique optimale π^* qui minimise la fonction cumulative de coût

Optimisation par Itération de la Politique

Nous utilisons une approche de **programmation dynamique** pour trouver une politique optimale.

- Nous commençons avec une politique initiale π_0 et une configuraton initiale s_0;
- Nous calculons une estimation $\overline{V}^{\pi_0}(s_0)$ de la fonction de coût cumulative;

- Nous améliorons progressivement la politique en utilisant l'estimation courante de la fonction cumulative coût:

$$\overline{V}^{\pi_t}(s) = \min_{s' \in S} \left(\delta(s, s') + \mathbb{E}\left[cost(s', q)\right] + \gamma \overline{V}^{\pi_{t-1}}(s') \right)$$

- Nous continuons l'amélioration jusqu'à ce qu'il n'y ait plus ou peu de changement de politique.

Difficultés dans l'Optimisation par Itération de la Politique

1. La **taille de l'espace des états** explose combinatoirement;
2. Il n'y a pas de modèle a priori de la **fonction de coût** $cost(s, q)$ et la **distribution des requêtes** est inconnue.

Solutions

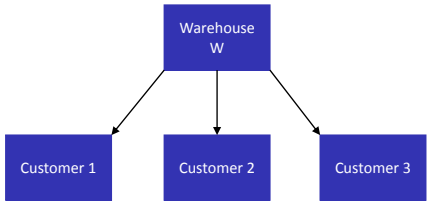
1. **Réduire la taille de l'espace des états**;

$$S_{s,\hat{q}} = \{s' \in S \mid cost(s, \hat{q}) > cost(s', \hat{q})\}$$

2. **Estimer itérativement la fonction de coût** grace à l'**algorithme des moindres carrés récursifs**.

COREIL

COREIL est un régulateur d' index qui décide de la création et de la destruction d' **index secondaires** (non-plaçants) en fonction de la charge de travail (requêtes) et de la stratégie d'apprentissage par renforcement presentée.



Queries
1) New order
2) Delivery
3) Stock

Tables
1) History
2) Stock
3) New orders
4) Stocks

Database Schema: **R**

Set of all Database Configurations: **S = {s}**

...	.....
t = 201	Customer 1, New order
t = 202	Stock
t = 203	Customer 2, Delivery
...	.....

Schedule of queries and updates: **Q**

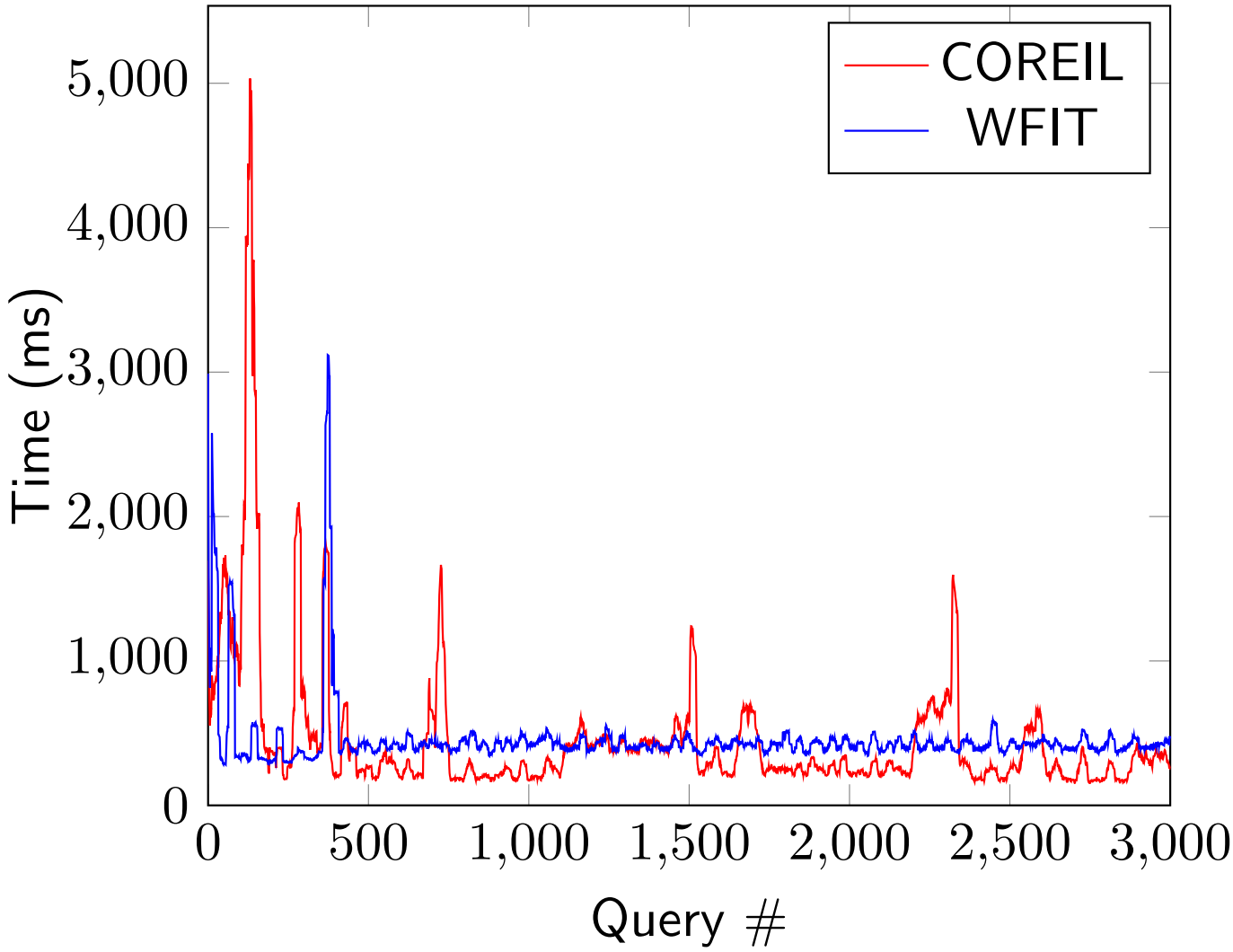
Evaluation: Jeu de Données et Charge de Travail

- Les résultats presentés ci-après sont obtenus avec le jeu de données et la charge de travail de référence de traitement en ligne des transactions TPC-C générés par l'outil OLTP-Bench;
  - Le facteur d'echelle est de 2; Chacun des 5 types de transactions est associé à 3 à 5 requêtes SQL (recherches et mises à jour);
- Le jeu de données est stocké et la charge de travail exécutée avec le système de gestion de bases de données DB2 d'IBM.

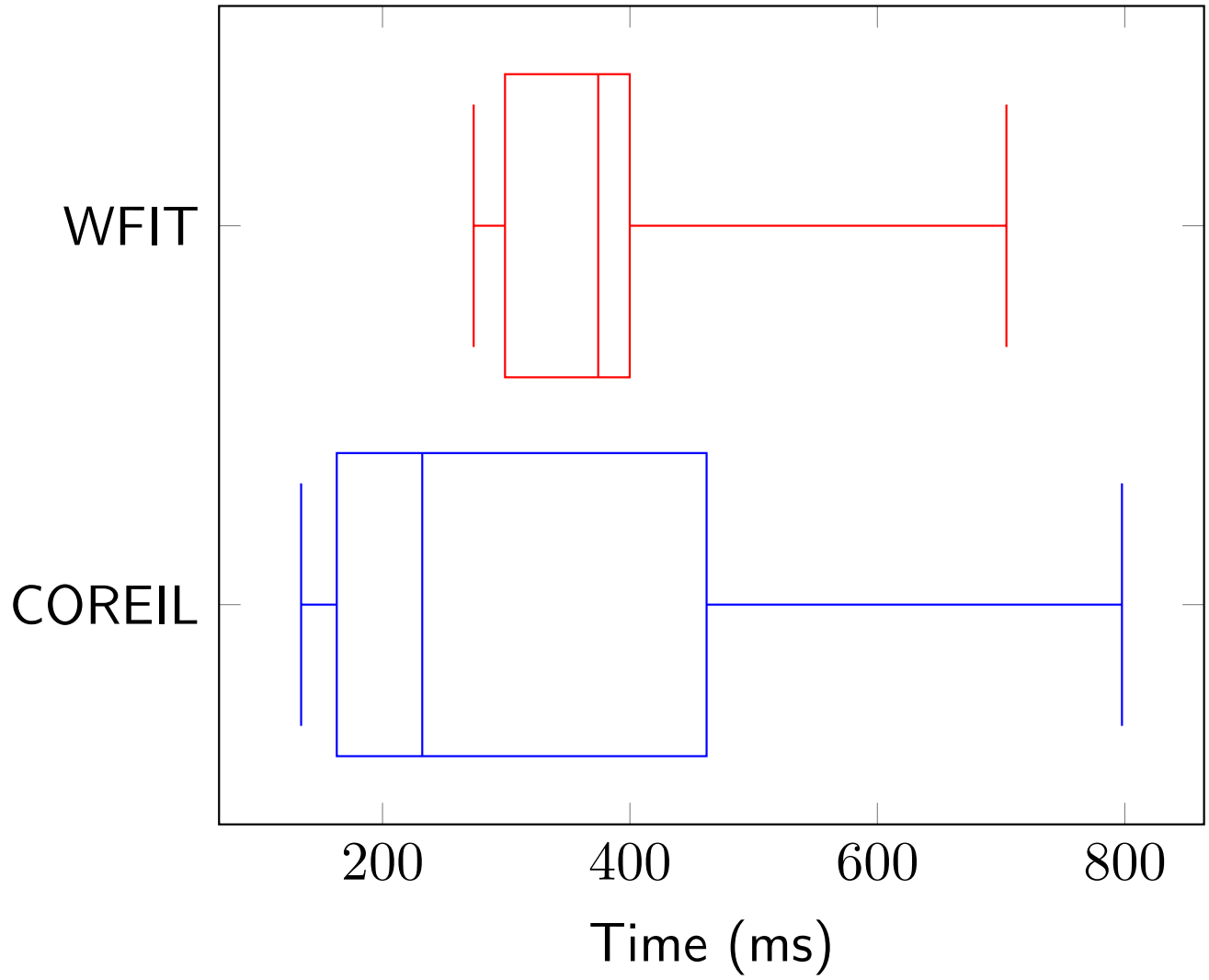
Evaluation: Référence et Métriques

- Nous comparons avec WFIT (VLDB 2012) une modélisation du problème comme un système métrique de tâches résolu par l'algorithme de la fonction de travail et utilsant le modèle de coût de l'optimiseur What-if de DB2.
- Nous mesurons l'efficacité, le coût supplémentaire, le coût de changement de configuration et l'effectivité.

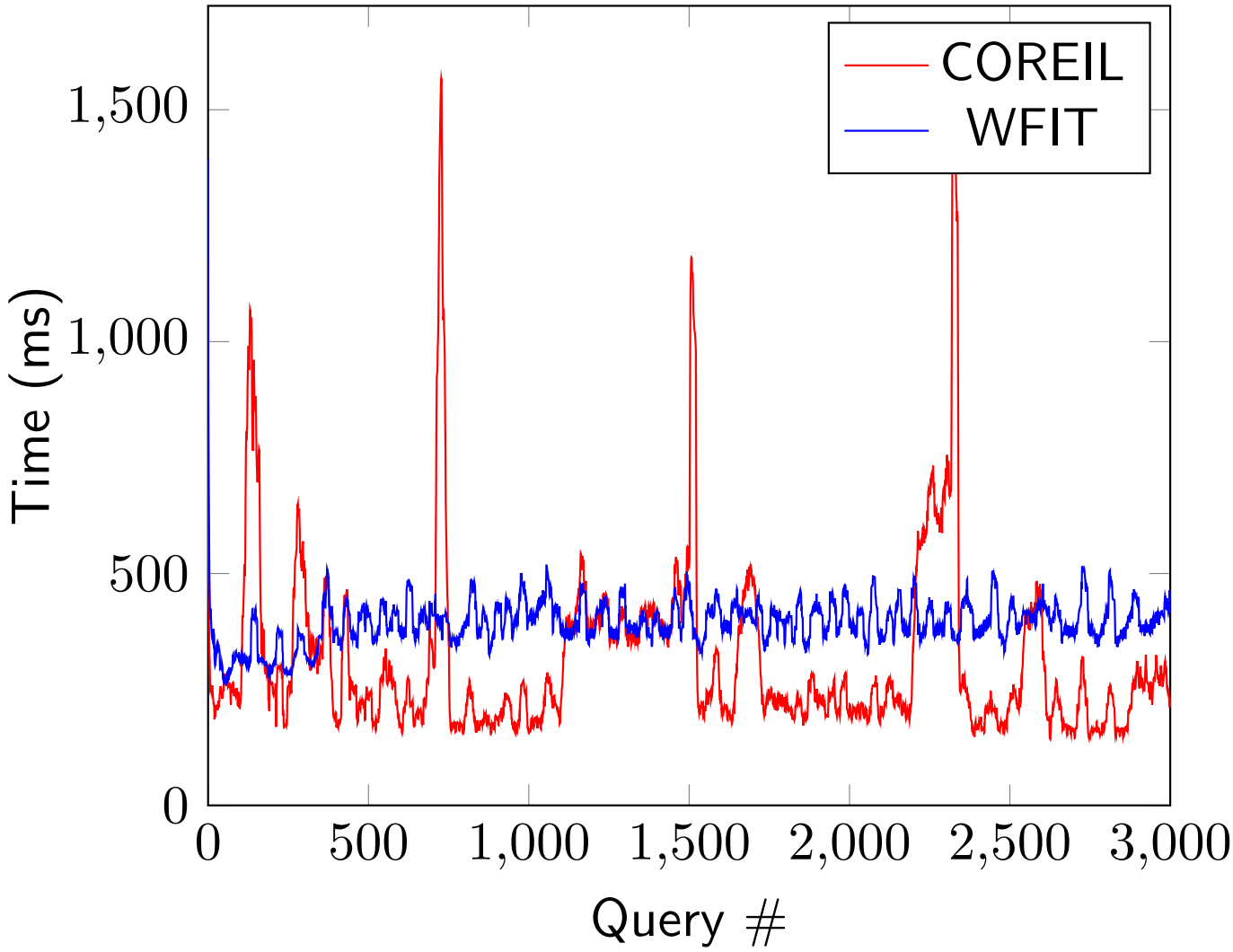
Efficacité


COREIL
WFIT
Time (ms)
5,000
4,000
3,000
2,000
1,000
0
0
500
1,000
1,500
2,000
2,500
3,000
Query #

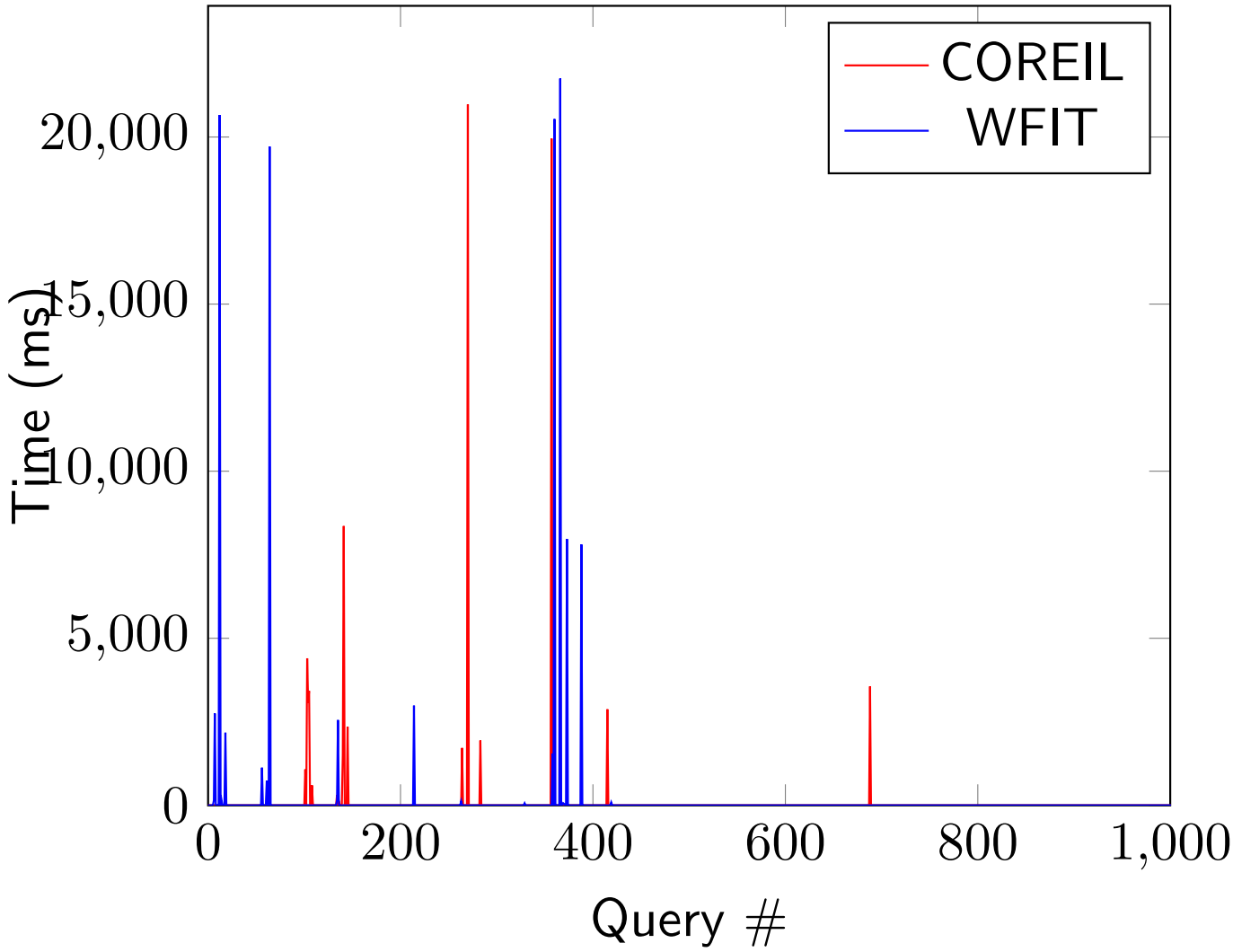
Boîte à Moustaches



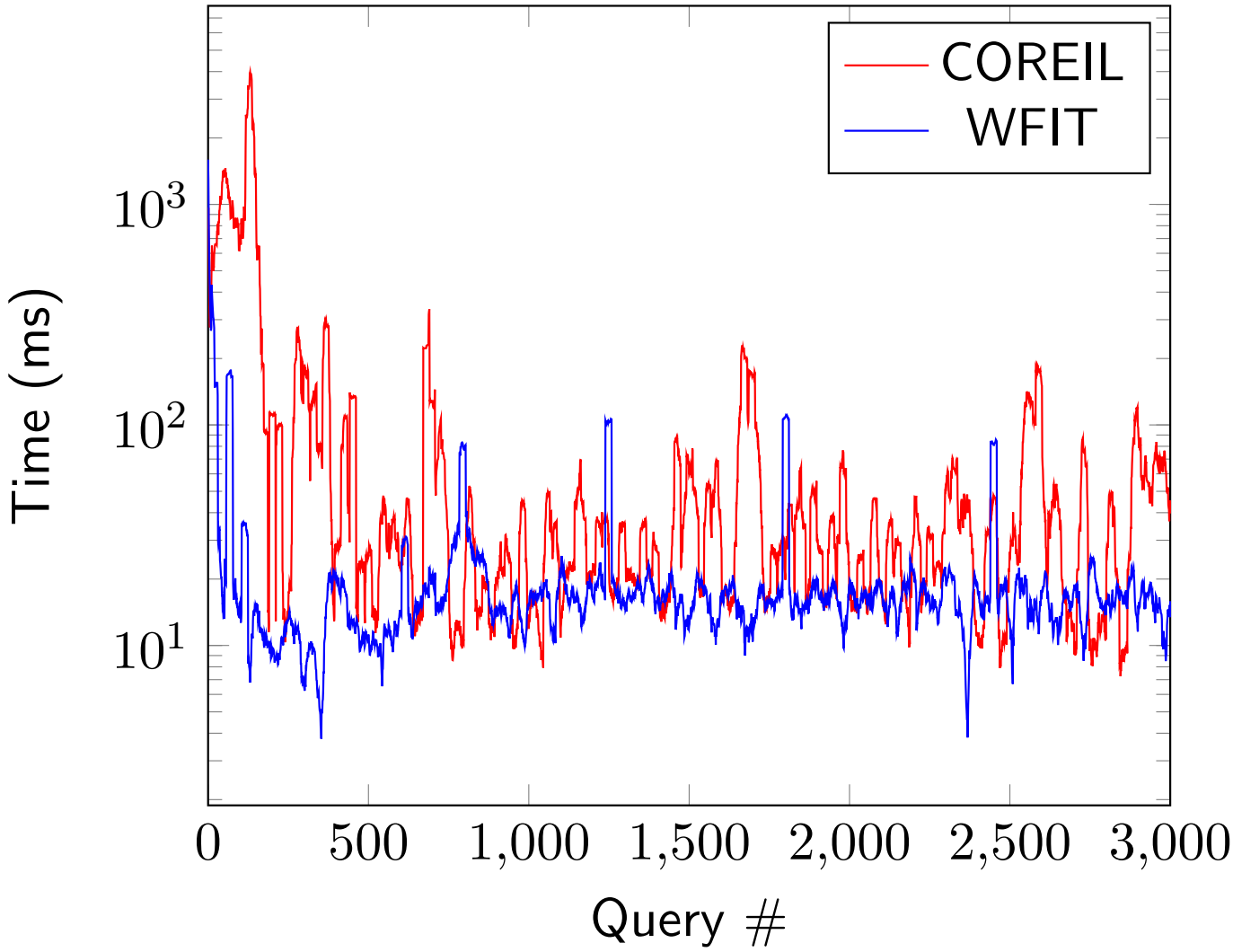
Coût Supplémentaire



Coût de Changement de Configuration



Effectivité



Conclusion

- Nous avons présenté une approche du **réglage des configurations** des systèmes modélisé par un **processus de décision Markovien** et optimisé avec un **apprentissage par renforcement**.
- Nous avons présenté **COREIL**, une application de cette approche au **réglage des index** dans les systèmes de gestion de bases de données.
- Comparé à l'état de l'art, **COREIL** est **efficace** et **effectif**.