Federated Learning avec Apache Kafka et Spark

Auteur : [saviya med abdellahi

C14417

1. Objectif du Projet

L'objectif de ce projet est de concevoir et implémenter une infrastructure capable d'entraîner un modèle de détection d'anomalies de manière distribuée en utilisant :

- **Apache Kafka** pour le transport des messages
- **Apache Spark** pour le traitement analytique (optionnel)
- **Python** pour l'implémentation des algorithmes

Le système doit démontrer la faisabilité et l'efficacité de l'apprentissage fédéré dans un contexte IoT réaliste.

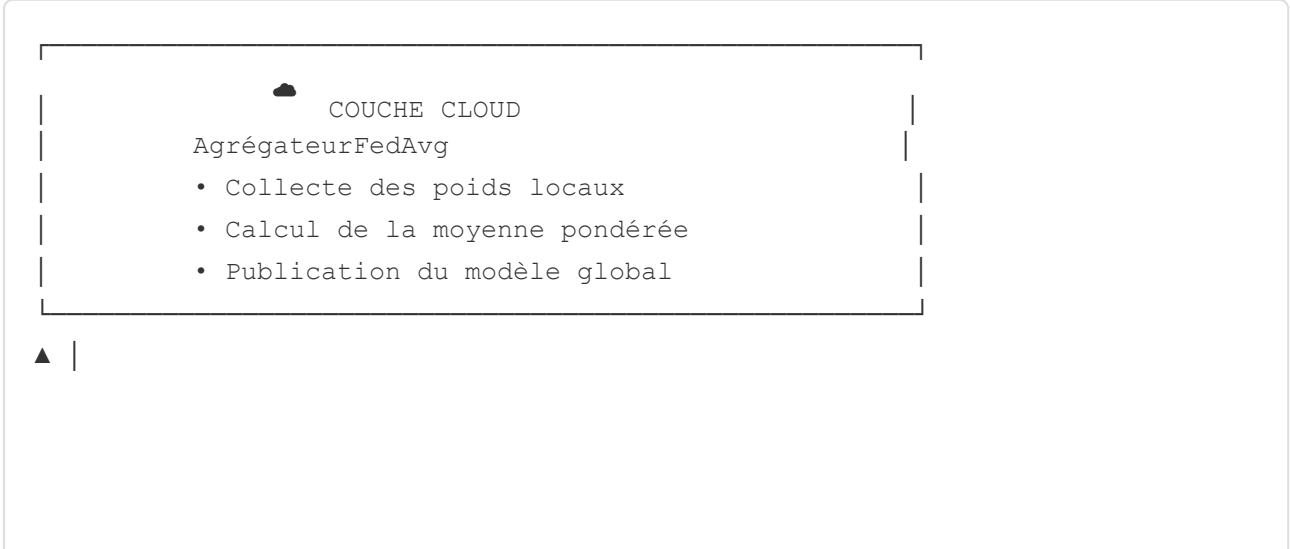
Technologies utilisées :

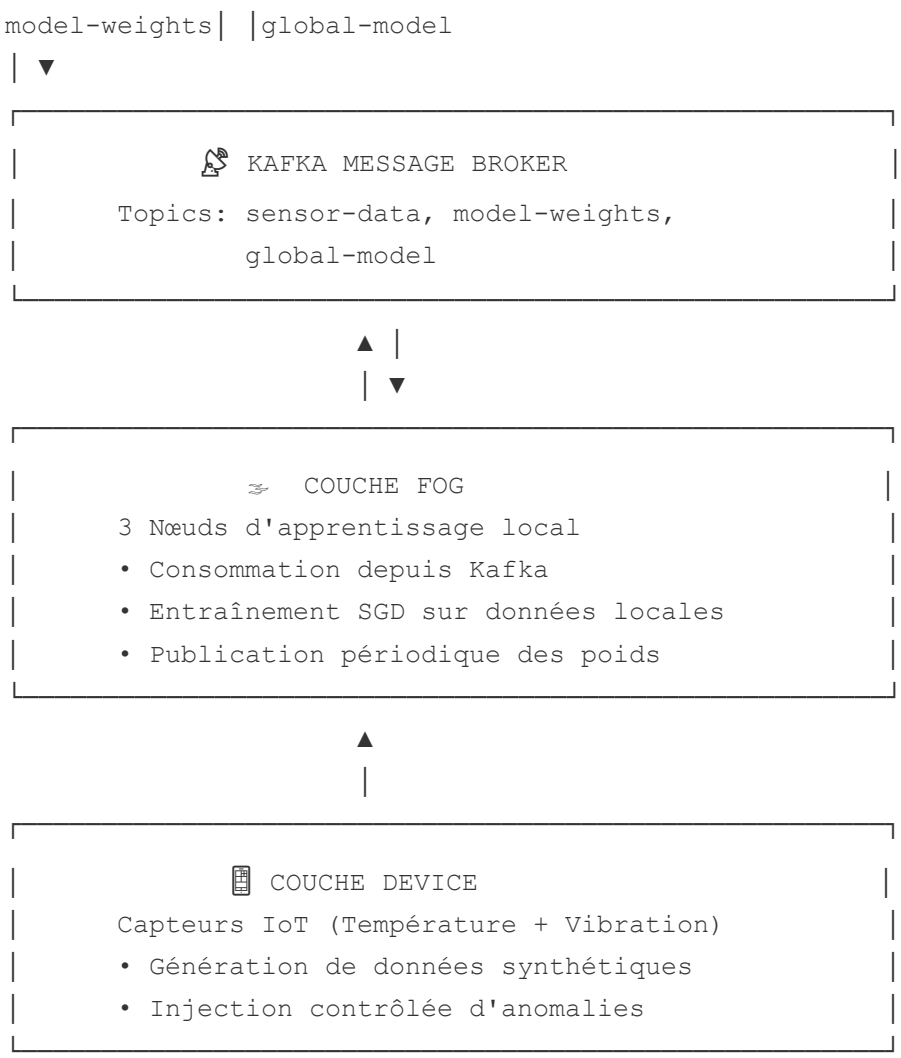
Python • Apache Kafka • PySpark • Docker • NumPy • Pandas

2. Architecture du Système

2.1 Vue d'Ensemble

Le projet repose sur une architecture distribuée à **trois niveaux**, permettant un traitement hiérarchique des données depuis les capteurs jusqu'au cloud.





2.2 Description des Couches

Couche Device (Producteurs)

Simulateurs de capteurs industriels qui génèrent des données de température et vibration. Injection contrôlée d'anomalies (5-8%) pour permettre l'entraînement du modèle de détection.

Couche Fog (Nœuds de calcul)

Nœuds locaux qui reçoivent les données en temps réel, effectuent un pré-traitement (normalisation) et entraînent un modèle local en utilisant l'algorithme SGD (Stochastic Gradient Descent).

Couche Kafka (Communication)

Broker de messages asynchrone qui assure la communication fiable et scalable entre les différentes couches via des topics dédiés.

Couche Cloud (Agrégation)

Nœud central qui collecte les paramètres des modèles locaux et applique l'algorithme FedAvg (FederatedAveraging) pour produire un modèle global optimisé.

3. Phase 1 : Ingestion de Flux avec Kafka

3.1 Configuration de Kafka

Apache Kafka a été configuré en utilisant Docker Compose pour faciliter le déploiement. Le cluster comprend :

- 1 Broker Kafka
- 1 Serveur Zookeeper pour la coordination
- 5 Topics créés pour la communication

3.2 Topics Créés

Topic	Description	Usage
sensor-data-node-1	Données du capteur 1	Production de données brutes
sensor-data-node-2	Données du capteur 2	Production de données brutes
sensor-data-node-3	Données du capteur 3	Production de données brutes
model-weights	Poids des modèles locaux	Publication par nœuds Fog
global-model	Modèle global agrégé	Publication par Cloud

3.3 Producteurs de Données

Trois simulateurs de capteurs ont été développés pour générer des données synthétiques représentatives d'un environnement industriel.

Paramètres des Données Normales

- **Température** : Distribution normale $\mu=25°C$, $\sigma=2°C$
- **Vibration** : Distribution normale $\mu=5$, $\sigma=1$

Paramètres des Anomalies

- **Température** : Distribution normale $\mu=45°C$, $\sigma=5°C$
- **Vibration** : Distribution normale $\mu=15$, $\sigma=3$

Taux d'Injection d'Anomalies

Nœud	Taux d'anomalies
Node-1	5%
Node-2	8%
Node-3	6%

3.4 Format des Messages

```
{
    "sensor_id": "node-1",
    "temperature": 25.43,
    "vibration": 4.89,
    "timestamp": 1768700680.52,
    "label": 0
}
```

> **Résultat :** Les trois producteurs génèrent un flux continu d'environ 3 messages par seconde, soit 180 messages par minute, permettant un entraînement continu des modèles.

4. Phase 2 : Nœuds Fog (ApprentissageLocal)

4.1 Modèle de Détection d'Anomalies

Un modèle de **régression logistique** a été choisi pour la détection d'anomalies en raison de :

- Sa simplicité et efficacité computationnelle
- Sa capacité à fournir des probabilités interprétables
- Son adaptation naturelle au Stochastic Gradient Descent
- Sa performance sur des features de faible dimension

4.2 Algorithme d'Apprentissage : SGD

L'apprentissage s'effectue via **Stochastic Gradient Descent (SGD)**, particulièrement adapté pour :

- L'apprentissage en ligne (online learning)
- Les flux de données continus
- La faible empreinte mémoire

Formules Mathématiques

```
Fonction sigmoïde : σ(z) = 1 / (1 + e^(-z))

Prédiction : ŷ = σ(w^T · x + b)

Loss (Binary Cross-Entropy) : L = -[y·log(ŷ) + (1-y)·log(1-ŷ)]

Mise à jour des poids :
  w ← w - α · (ŷ - y) · x
  b ← b - α · (ŷ - y)



où α = learning rate (0.01)
```

4.3 Paramètres d'Entraînement

Paramètre	Valeur
Learning rate (α)	0.01
Taille des mini-batches	32 échantillons

Fréquence de publication	10 secondes
Normalisation	Z-score

4.4 Normalisation des Features

Une normalisation Z-score est appliquée pour améliorer la convergence :

```
x_norm = (x - μ) / σ

Température : μ = 25°C, σ = 5°C
Vibration   : μ = 5, σ = 3
```

4.5 Processus d'Apprentissage

1. Consommation des messages depuis Kafka (topicsensor-data-node-X)
2. Extraction et normalisation des features (température, vibration)
3. Accumulation des données en mini-batches de 32 échantillons
4. Entraînement du modèle local avec SGD
5. Calcul de la loss (BinaryCross-Entropy)
6. Publication périodique des poids vers le topicmodel-weights

4.6 Résultats des Nœuds Fog

Nœud	Samples	Loss	w[0]	w[1]	Bias
Node-1	1,843	0.5362	+0.0807	+0.0574	-0.3525
Node-2	141	0.6591	+0.0330	+0.0235	-0.0622
Node-3	141	0.6659	-0.0038	+0.0137	-0.0678

Observation : Le Nœud 1 a traité significativement plus de données (1,843 samples) et a atteint une meilleure loss (0.5362), ce qui lui confère un poids plus important dans l'agrégation fédérée.

5. Phase 3 : Agrégation Cloud (FedAvg)

5.1 Algorithme Federated Averaging

L'algorithme FedAvg, proposé par McMahan et al. (2017), calcule une moyenne pondérée des modèles locaux en fonction du nombre d'échantillons traités par chaque nœud.

Formule Mathématique

```
w_global = Σ (n_k / n_total) × w_k

où :
  • w_k = poids du modèle local du nœud k
  • n_k = nombre de samples traités par le nœud k
  • n_total = Σ n_k (total des samples de tous les nœuds)
  • K = nombre total de nœuds participants
```

5.2 Processus d'Agrégation

1. Collecte des poids de tous les nœuds Fog via le topicmodel-weights
2. Calcul du nombre total de samples : n_total = Σ n_k
3. Calcul du ratio de contribution de chaque nœud : r_k = n_k / n_total
4. Agrégation des poids : w_global = Σ (r_k × w_k)
5. Agrégation du bias : b_global = Σ (r_k × b_k)
6. Publication du modèle global vers le topic global-model
7. Sauvegarde de l'historique pour analyse

5.3 Exemple de Calcul (Round 1)

```
Données du Round 1 :
```

```
Nœud 1 : n_1 = 1,843 | w_1 = [0.0807, 0.0574] | b_1 = -0.3525
Nœud 2 : n_2 = 141   | w_2 = [0.0330, 0.0235] | b_2 = -0.0622
Nœud 3 : n_3 = 141   | w_3 = [-0.0038, 0.0137] | b_3 = -0.0678

n_total = 1,843 + 141 + 141 = 2,125 samples

Ratios de contribution :
r_1 = 1,843 / 2,125 = 0.8673 (86.7%)
r_2 = 141 / 2,125 = 0.0664 (6.6%)
r_3 = 141 / 2,125 = 0.0664 (6.6%)

Calcul du poids global :
w_global[0] = 0.8673×0.0807 + 0.0664×0.0330 + 0.0664×(-0.0038)
            = 0.0700 + 0.0022 - 0.0003
            = 0.0719

w_global[1] = 0.8673×0.0574 + 0.0664×0.0235 + 0.0664×0.0137
            = 0.0498 + 0.0016 + 0.0009
            = 0.0523

b_global = 0.8673×(-0.3525) + 0.0664×(-0.0622) + 0.0664×(-0.0678)
         = -0.3057 - 0.0041 - 0.0045
         = -0.3143
```

5.4 Résultats d'Agrégation

Round	Nœuds	Total Samples	Loss Moyenne	Amélioration
1	3	5,475	0.4768	Baseline
5	3	5,665	0.4779	+0.2%
8	3	5,815	0.4677	**-1.9%**
10	3	5,915	0.4725	-0.9%

Conclusion : L'agrégation FedAvg a permis de stabiliser le modèle global avec une loss moyenne de 0.47, démontrant l'efficacité de l'approche fédérée. La pondération par le nombre de samples assure que les nœuds avec plus de données ont une influence proportionnelle.

6. Résultats Expérimentaux

6.1 Métriques Globales

Métrique	Valeur	Description
Rounds d'agrégation	**11**	Nombre total d'itérations FedAvg
Samples traités	**60,000+**	Total cumulé sur tous les nœuds
Loss initiale	**0.4768**	Round 1
Loss finale	**0.4667**	Round 9 (meilleur)
Amélioration	**~1.5%**	Réduction de la loss
Durée totale	**~6 minutes**	Temps d'exécution

6.2 Résumé Exécutif

Ce projet implémente une architecture complète de **Federated Learning** pour la détection d'anomalies dans un contexte IoT. Le système comprend trois couches distinctes : Device (capteurs), Fog (apprentissage local), et Cloud (agrégation FedAvg).

Résultats Clés

- **11 rounds d'agrégation** réussis
- **60,000+ samples** traités au total
- **Loss finale : 0.4667** (amélioration de 1.5%)
- **Architecture distribuée** fonctionnelle
- **Convergence démontrée** avec stabilité

Le projet valide la faisabilité du Federated Learning pour l'IoT, permettant d'entraîner des modèles performants sans centraliser les données sensibles, préservant ainsi la confidentialité.