A Parallel Random Forest Algorithm for Big Data in a Spark Cloud Computing Environment

Fiche de lecture

Supervisé par :
Dr. MESSI

Faculté des Sciences

Nom :	**DJIEMBOU TIENTCHEU VICTOR NICO**
Devoir :	2
Publié à :	IEEE Transactions on Parallel and Distributed Systems, 2016
Auteur :	Jianguo Chen, Kenli Li et Al.
Mots clés :	Apache Spark, Big Data, Cloud Computing Data Parallel, Random Forest, Task Parallel
UE :	INF5099
Matricule :	17T2051
Département :	Informatique
Niveau - Option :	M2 - Sciences des Données

Table des matières

1 Contexte

Le domaine de la fouille de données et même de l'analyse de gros volumes de données intervient lorsque nous faisons face à un très grand spectre d'information à manipuler dans une domaine métier. Wu, X., Zhu et Al. dans **Data mining with big data** présentent un gonflement double du taux de données généré en continu chaque deux ans. Alors, les études basées sur la distribution et la parallélisation sur le cloud de processus de fouille de données est devenu un sujet en vogue pour les chercheurs et les entreprise qui ont progressivement proposé des solutions.

Hadoop est un une plateforme cloud très étendu pour l'analyse de très grand volume de données. En son sein sont implémentés de nombreux modèle de machine learning basé sur **MapReduce**. Toutefois, les ses implémentation sont sujet à des génération de sorties intermédiaire à chaque itération qui sont sous format Hadoop Distributed File System (HDFS) et donc doivent être chargé depuis le disque. Ce chargement a un coût considérable en terme d'entrer **I/O**.

Apache Spark est une autre solution cloud qui s'inscrit dans le même cadre que la plateforme Hadoop. Apache est dotée de deux atouts majeurs, un model de resilience distribuée des données (RDD) et un modèle de Graphe Acyclique Orienté (DGA) qui permettent faciliter l'embarquement de données sur une approche de calcul en mémoire. De ce fait, Apache possède déjà une plus pour les solutions de distribution et parallélisation d'algorithme de data mining sur de grand volume de données.

Alors, ayant prise en compte l'existant, ils proposent une approche détaillée de la logique de parallélisation des forêts aléatoires dans un environnement de calcul en nuage Spark pour le traitement de gros ensembles de données.

2 Problème

Le traitement de gros ensembles de données pose des défis en termes de temps d'exécution et de gestion des ressources. L'objectif est de développer une méthode efficace pour construire des RF sur ces ensembles de données massifs tout en maintenant une précision élevée.

3 motivations

Les RF sont des modèles d'apprentissage automatique populaires et puissants, mais leur construction séquentielle peut être lente pour les gros ensembles de données. La parallélisation des RF dans un environnement Spark permet d'accélérer le processus d'apprentissage tout en exploitant les capacités de calcul distribué.

4 Spark vs Hadoop

Spark est une plate-forme de calcul distribué qui offre des fonctionnalités avancées pour le traitement massif de données. Comparé à Hadoop, Spark se distingue par sa vitesse de traitement accrue grâce à la gestion de la mémoire distribuée et à l'optimisation du calcul en mémoire.

5 Parallélisation et distribution sur Spark

Spark permet la distribution automatique des tâches et la gestion de la mémoire distribuée, ce qui facilite la parallélisation des RF. Les tâches peuvent être réparties sur différents nœuds de calcul dans le cluster, optimisant ainsi l'utilisation des ressources disponibles.

6 Parallélisation de RF (PRF) sur Spark

L'algorithme PRF propose une approche spécifiquement conçue pour la parallélisation des RF dans un environnement Spark. Il divise l'ensemble de données en partitions plus petites et construit des RF en parallèle sur différents nœuds de calcul. L'agrégation des résultats permet d'obtenir une prédiction finale.

Le PRF repose sur la division des données, la construction parallèle des RF sur chaque nœud de calcul, l'utilisation de sous-ensembles aléatoires d'attributs et l'agrégation des résultats. Cette conception exploite les fonctionnalités de Spark pour obtenir des performances améliorées avec des ensembles de données massifs.

1. Division de l'ensemble de données : L'ensemble de données initial est divisé en plusieurs partitions plus petites. Chaque partition est attribuée à un nœud de calcul dans le cluster Spark. La division peut être réalisée selon différentes stratégies, telles que la division équitable basée sur la taille des partitions ou la division aléatoire.
2. Construction des arbres de décision en parallèle : Sur chaque nœud de calcul, une forêt aléatoire est construite en parallèle. Chaque nœud de calcul reçoit une ou plusieurs partitions de données et construit un ensemble d'arbres de décision basés sur ces partitions. Cela permet d'exploiter pleinement la puissance de calcul distribuée en exécutant la construction de chaque arbre de décision de manière simultanée et indépendante sur différents nœuds.
3. Parallélisme au niveau des arbres de décision : La construction de chaque arbre de décision peut également être parallélisée au niveau des attributs (features). Au lieu de construire l'arbre en explorant séquentiellement chaque attribut, des sous-ensembles aléatoires d'attributs peuvent être sélectionnés pour chaque nœud de calcul. Cela permet d'accélérer la construction de chaque arbre de décision en parallèle.
4. Agrégation des résultats : Une fois que tous les arbres de décision ont été construits, les résultats sont agrégés pour obtenir une prédiction finale. Dans le cas d'une classification, une méthode de vote majoritaire est généralement utilisée, où chaque arbre de décision contribue à la prédiction finale en votant pour la classe majoritaire. Pour la régression, les prédictions des différents arbres peuvent être moyennées pour obtenir la prédiction finale.

La plate-forme Spark offre des fonctionnalités de gestion automatique de la distribution des tâches et de la mémoire, ce qui facilite l'implémentation de la parallélisation des forêts aléatoires. Spark utilise une architecture de traitement distribué basée sur le modèle de programmation MapReduce, où les tâches sont distribuées automatiquement sur les nœuds de calcul disponibles dans le cluster. Cela permet d'optimiser l'utilisation des ressources disponibles et d'accélérer le processus de construction des arbres de décision.

L'utilisation de la parallélisation dans un environnement Spark permet de traiter efficacement des ensembles de données massifs. Les performances de l'algorithme parallèle sont évaluées en termes de temps d'exécution et de précision des prédictions par rapport à une implémentation séquentielle. Les résultats expérimentaux montrent généralement une amélioration significative des performances, avec des temps d'exécution réduits et une précision comparable à celle de l'approche séquentielle.

7 Conclusion

la parallélisation des RF sur Spark offre une solution prometteuse pour l'analyse de gros ensembles de données en exploitant les capacités de calcul distribué de Spark. Le PRF, spécifiquement conçu pour cette tâche, permet d'accélérer le processus d'apprentissage tout en maintenant une précision élevée. Cette approche ouvre de nouvelles opportunités pour le traitement efficace des données massives dans le domaine de l'apprentissage automatique.

Références

[Apa] Apache, *Hadoop*, Website.

[WZWD13] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding, *Data mining with big data*, IEEE transactions on knowledge and data engineering **26** (2013), no. 1, 97–107.