گزارش تمرین سوم - بخش دوم

پریا مهدیان 9931079

سوال1:

تفاوت های معماری:

RDD ها یا Resilient Distributed Datasets (مجموعه داده های توزیع شده انعطاف پذیر)، انتزاع داده های بنیادی یا فاندامنتال Spark هستند که مجموعه ای از عناصر تقسیم بندی شده در گره های یک کلاستر را نشان می دهند. آنها مجموعه ای از اشیاء توزیع شده در سطح پایین و تغییرناپذیر را فراهم می کنند. RDD ها داده های واقعی را به همراه اطلاعات اصل و نسب ذخیره می کنند تا پارتیشن های داده از دست رفته را بازسازی کنند.

DataFrames یک انتزاع سطح بالاتری است که بر روی RDD ها ساخته شده است و برای دستکاری آسان تر داده ها طراحی شده است. آنها داده ها را در ستون های نامگذاری شده سازماندهی می کنند، شبیه به یک جدول در یک پایگاه داده رابطه ای، و بیشتر SQL گرا هستند. DataFrames در Spark از بهینه‌سازی‌هایی مانند Catalyst Query Optimizer و موتور اجرای تنگستن برای عملکرد بهتر استفاده می‌کند.

تفاوت های عملکردی:

درRDDها کاربران کنترل دقیقی بر روی داده ها و محاسبات دارند. تبدیل های RDD صریح هستند و نیاز به مشخص کردن نحوه تبدیل داده ها دارند. ایده‌آل برای تبدیل‌های سطح پایین، دستکاری پیچیده داده‌ها و زمانی که به کنترل دقیق روی داده‌ها نیاز است. مناسب برای پردازش داده های بدون ساختار یا هنگام برخورد با داده هایی که با فرمت جدولی مناسب نیستند.

در مقابل DataFrameها یک API شفاف تر و سطح بالاتر ارائه می دهد که محاسبات پیچیده را در زیر hood بهینه می کند. آنها عملکرد بهینه را از طریق بهینه سازی Catalyst ارائه می دهند و برای عملیات مشابه SQL مناسب تر هستند. برای پردازش داده های ساختاریافته، پرس و جوهای SQL، و زمانی که بهینه سازی عملکرد مانند pushdown گزاره و هرس ستون ضروری است، ترجیح داده می شود. دیتا فریم ها معمولاً به دلیل بهینه سازی Catalyst و تولید کد، عملکرد بهتری نسبت به RDD ها ارائه می دهند.

موارد استفاده:

RDDها ایده‌آل هستند برای تبدیل‌های سطح پایین، دستکاری پیچیده داده‌ها و زمانی که به کنترل دقیق روی داده‌ها نیاز است. آن ها مناسب برای پردازش داده های بدون ساختار یا هنگام برخورد با داده هایی که با فرمت جدولی مناسب نیستند.

DataFrameها برای پردازش داده های ساختاریافته، پرس و جوهای SQL، و زمانی که بهینه سازی عملکرد مانند pushdown گزاره و هرس ستون ضروری است، ترجیح داده می شود. دیتا فریم ها معمولاً به دلیل بهینه سازی Catalyst و تولید کد، عملکرد بهتری نسبت به RDD ها ارائه می دهند.

سوال2:

قابلیت های Query Capabilities یا پرس و جو: Spark SQL به کاربران اجازه می دهد تا درخواست های SQL را مستقیماً روی DataFrames یا جداول با استفاده از سینتکس SQL بنویسند. DataFrame API یک روش برنامه ریزی شده برای دستکاری و پردازش داده ها ارائه می دهد.

بهینه سازی عملکرد: پرس و جوهای SQL توسط بهینه ساز Spark Catalyst، شبیه به عملیات DataFrame بهینه سازی می شوند. SQL Spark را می توان برای کاربران آشنا با سینتکس SQL ترجیح داد، در حالی که DataFrame API برای دستکاری داده های پیچیده انعطاف پذیرتر است.

قابلیت استفاده DataFrame API: API برای توسعه دهندگانی که با برنامه نویسی در زبان هایی مانند Python، Scala یا Java آشنا هستند، انعطاف پذیرتر و بصری تر است. از طرف دیگر، SQL Spark برای کاربرانی که با پرس و جوی SQL راحت هستند مناسب است.

سوال3:

پارتیشن بندی داده ها در Spark شامل تقسیم داده ها به تکه های توزیع شده در گره های یک کلاستر است. برای دستیابی به موازی سازی و پردازش کارآمد داده در محیط های توزیع شده بسیار مهم است. پارتیشن بندی بر محل داده ها، زمان بندی کارها و عملکرد کلی کار با به حداقل رساندن به هم زدن داده ها تاثیر می گذارد.

سوال4:

Hash Partitioning: داده ها بر اساس مقدار هش یک ستون تقسیم بندی می شوند. برای توزیع برابر داده ها و اتصالات روی یک کلید مفید است.

Range Partitioning: داده ها بر اساس محدوده های خاصی از مقادیر تقسیم بندی می شوند. مناسب برای پرس و جوهای محدوده و داده های مرتب شده است.

Custom Partitioning: به کاربران اجازه می دهد تا منطق پارتیشن بندی سفارشی را بر اساس نیازهای خاص خود تعریف کنند.

تاثیر: پارتیشن بندی مناسب می تواند عملکرد شغلی را با افزایش data locality، کاهش به هم زدن داده ها(shuffeling) در طول تبدیل ها و فعال کردن پردازش موازی بهبود بخشد. استراتژی های پارتیشن بندی ناکارآمد می تواند منجر به انحراف داده ها و استفاده غیر بهینه از منابع شود.