

دانشگاه صنعتي امیرکبیر  
(پلی تکنیک تهران)

دانشكده مهندسی کامپیوتر

تمرین سوم درس رایانش ابری – بخش تشریحی

نگارش

بردیا اردکانیان، آراد فیروزکوهی

استاد

دکتر جوادی

بهمن ۱۴۰۱

**بخش اول)**

**مفهوم replication چیست؟**

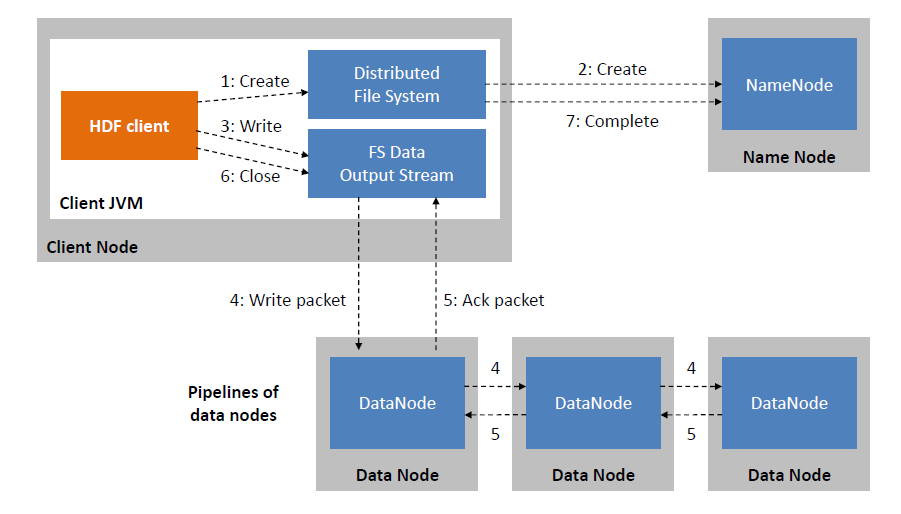
فرآیندی است که در آن داده‌های موجود در سرور(های) فیزیکی/مجازی یا سرویس‌های ابری به طور مداوم در سرور(های) ثانویه یا سرویس‌های ابری دیگر تکثیر یا کپی می شوند تکثیر داده (data replication) می‌گویند.

سازمان‌ها داده‌ها را برای پشتیبانی از دسترسی بالا، بکاپ گیری و/یا رکاور کردن از شکست تکثیر می‌کنند. HDFS به گونه‌ای طراحی شده است که فایل های بسیار بزرگ را به طور قابل اعتماد در سراسر ماشین‌ها در یک خوشه بزرگ ذخیره می‌کند. هر فایل را به صورت دنباله‌ای از بلوک ها ذخیره می‌کند. همه بلوک های یک فایل به جز آخرین بلوک یک اندازه هستند. بلوک های یک فایل برای تحمل خطا تکرار می شوند. به عنوان مثال اگر داده‌ ما به سه بلاک تقسیم شود و replication factor ما ۳ باشد از هر بلاک ۳ کپی در سراسر سیتسم ذخیره سازی می‌کند که اگر بعضی از سرور‌ها از دسترس خارج شوند داده‌ها به سادگی قابل بازیابی داشته باشند. جدول زیر می‌تواند محتوای namenode و datanode باشد.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Rack3 | Rack2 | Rack1 |  | Namenode |
|  | B2 | B1 |  | Datanode #1 |
| B1 | B3 |  |  | Datanode #2 |
| B3 |  | B2 |  | Datanode #3 |

**مفهموم block در HDFS چیست و اگر بلاک‌ها را بسیار کوچک در نظر بگیریم چه مشکلی پیش می‌آید؟**

HDFS فایل های بزرگ را به تکه های کوچک تقسیم می کند که به آن بلوک می گویند. اینها کوچکترین واحد داده در سیستم فایل هستند. NameNode تصمیم خواهد گرفت که داده ها در کجای DataNode ذخیره شوند. تمام بلوک فایل ها به جز آخرین بلوک، اندازه یکسانی دارند. در Apache Hadoop، اندازه بلوک پیش فرض ۶۴ مگابایت است. Block حداقل مقدار داده ای است که می تواند بخواند یا بنویسد. بلوک‌های HDFS به طور پیش فرض ۶۴ مگابایت هستند و این قابل تنظیم است. HDFS فایل‌ها را به n تکه‌ با اندازه بلوک تقسیم می‌کند که به عنوان واحدهای مستقل ذخیره می‌شوند.



اگر بلوک‌ها کوچک باشند، بلوک‌های زیادی در Hadoop HDFS وجود خواهد داشت و بنابراین metadata زیادی برای ذخیره‌سازی وجود خواهد داشت. مدیریت چنین تعداد زیادی بلوک و metadata باعث ایجاد سربار و منجر به ترافیک در یک شبکه می شود. همچنین در صورتی که نیاز به خواندن داده‌ای باشیم که در datanode فعلی وجود ندارد باید از طریق شبکه داده را بخوانیم. اگر اندازه بلوک‌ها خیلی کوچک باشد و داده‌ها پخش شوند مجبور می‌شویم مدام از شبکه بخوانیم که سرعت را بسیار پایین می‌آورد.

**بخش دوم)**

**مزیت اسپارک نسبت به مدل قبلی map/reduce چیست؟**

تفاوت اصلی Spark و MapReduce در این است که Spark داده‌ها را برای مراحل بعدی پردازش کرده و در حافظه نگه می دارد، در حالی که MapReduce داده ها را روی دیسک پردازش می‌کند. در نتیجه، برای بارهای کاری کوچکتر، سرعت پردازش داده Spark تا 100 برابر سریعتر از MapReduce است.

علاوه بر این، برخلاف فرآیند اجرای دو مرحله‌ای درMapReduce، Spark یک گراف غیر چرخه‌ای جهت‌دهی شده (DAG) برای زمان‌بندی وظایف و هماهنگ‌سازی گره‌ها در سراسر خوشه Hadoop ایجاد می‌کند. این فرآیند ردیابی کار، تحمل خطا را امکان پذیر می کند، که عملیات ثبت شده را مجدداً روی داده های یک وضعیت قبلی اعمال می‌کند.

همچنین در اسپارک وضعیت فعلی به صورت RDD ذخیره می‌شود و action و transformation‌های انجام شده توسط لاگ ذخیره می‌شوند تا اگر جایی در مراحل دچار خطا شویم تنها مرحله از دست رفته را دوباره انجام دهیم. در صورتی که در MapReduce چون روی حافظه می‌نویسیم مرحله قبلی در دیسک ذخیره شده و کل مرحله قبلی را از اول انجام می‌دهد.

تفاوت های کلیدی Hadoop و Spark

* عملکرد: Spark سریعتر است زیرا از حافظه دسترسی تصادفی (RAM) به جای خواندن و نوشتن داده های میانی روی دیسک استفاده می کند. Hadoop داده ها را در چندین منبع ذخیره می کند و آنها را به صورت دسته ای از طریق MapReduce پردازش می کند.
* هزینه: Hadoop با هزینه کمتری اجرا می شود زیرا برای پردازش داده ها به هر نوع ذخیره سازی دیسک متکی است. Spark با هزینه بالاتری اجرا می شود زیرا برای پردازش داده های بلادرنگ به محاسبات درون حافظه متکی است، که نیاز به استفاده از مقادیر بالای RAM برای چرخش گره ها دارد.
* پردازش: اگرچه هر دو پلتفرم داده ها را در یک محیط توزیع شده پردازش می کنند، Hadoop برای پردازش دسته ای و پردازش داده های خطی ایده آل است. Spark برای پردازش بلادرنگ و پردازش جریان های داده های بدون ساختار زنده ایده آل است.
* مقیاس پذیری: زمانی که حجم داده ها به سرعت رشد می کند، Hadoop به سرعت برای پاسخگویی به تقاضا از طریق سیستم فایل توزیع شده Hadoop (HDFS) مقیاس می شود. به نوبه خود، Spark برای حجم زیادی از داده به HDFS مقاوم در برابر خطا متکی است.
* امنیت: Spark امنیت را با احراز هویت از طریق راز مشترک یا ثبت رویداد افزایش می‌دهد، در حالی که Hadoop از چندین روش تأیید هویت و کنترل دسترسی استفاده می‌کند. اگرچه، به طور کلی، Hadoop امن تر است، Spark می تواند با Hadoop ادغام شود تا به سطح امنیتی بالاتری برسد.
* یادگیری ماشینی (ML): Spark پلتفرم برتر در این دسته است زیرا شامل MLlib است که محاسبات تکراری ML در حافظه را انجام می دهد. همچنین شامل ابزارهایی است که رگرسیون، طبقه بندی، ماندگاری، ساخت خط لوله، ارزیابی و غیره را انجام می دهند.

**تفاوت action و transform در اسپارک چیست؟**

Transformation: تبدیل هر عملیات Spark است که DataFrame،Dataset یا RDD را برمی گرداند. وقتی زنجیره‌ای از تبدیل‌ها را ایجاد می کنیم، بلوک های ساختمانی را به کار Spark اضافه می‌کنیم، اما هیچ داده ای پردازش نمی‌بشود. این امکان پذیر است زیرا تبدیل‌ها به تنبلی اجرا می‌شوند. اسپارک در صورت لزوم مقدار را محاسبه می‌کند. البته، این بدان معناست که Spark باید مقادیر را مجدداً محاسبه کند، زمانی که ما از همان تبدیل‌ها دوباره استفاده می‌کنیم. ما می توانیم با استفاده از توابع persist یا cache از آن جلوگیری کنیم.

Action: به صورت lazy انجام نمی‌شوند. وقتی عملی را در کد قرار می‌دهیم و Spark هنگام اجرای کار به آن خط کد می‌رسد، باید تمام تبدیل‌هایی را که منجر به آن عمل می‌شود برای تولید یک مقدار انجام دهد. در اینجا تولید ارزش مفهوم کلیدی است. در حالی که تبدیل‌ها یکی از انواع داده Spark را برمی‌گردانند، اکشن‌ها تعدادی از عناصر (مثلاً تابع شمارش)، فهرستی از آنها (جمع‌آوری، گرفتن و غیره) را برمی‌گردانند یا داده‌ها را در حافظه خارجی ذخیره می‌کنند (نوشتن، saveAsTextFile، و دیگران).

تفاوت:

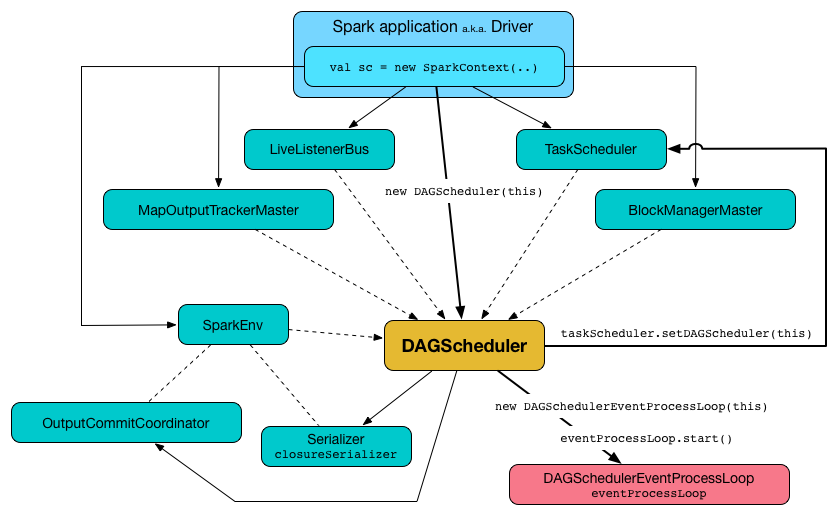
وقتی به Spark API نگاه می کنیم، به راحتی می توانیم تفاوت بین تبدیل ها و اقدامات را تشخیص دهیم. اگر تابعی DataFrame، Dataset یا RDD را برمی‌گرداند، یک تبدیل است. اگر چیز دیگری را برمی گرداند یا اصلاً مقداری را برمی گرداند (یا در مورد Scala API Unit را برمی گرداند)، یک عمل است.

**اسپارک شافل)**

**DAG Scheduler در اسپارک چیست؟**

لایه زمان‌بندی سطح بالا که زمان‌بندی مرحله‌گرا را اجرا می‌کند. این یک DAG از مراحل را برای هر کار محاسبه می کند، RDD ها و خروجی های مرحله تحقق یافته را ردیابی می کند، و حداقل زمانبندی برای اجرای کار پیدا می کند.

DAGScheduler لایه زمانبندی آپاچی اسپارک است که برنامه ریزی مرحله محور را با استفاده از Jobs و Stage پیاده سازی می کند. DAGScheduler یک طرح اجرای منطقی (تبدیل وابستگی های RDD ساخته شده با استفاده از تبدیل های RDD) را به یک طرح اجرای فیزیکی تبدیل می کند.



**گزارش HDFS)**

**coalesce وrepartition در اسپارک چیستند و کاربر اصلی آنها چیست؟**

Spark داده ها را به پارتیشن ها تقسیم می کند و محاسبات را روی پارتیشن ها به صورت موازی اجرا می کند.

Coalesce: روش coalesce تعداد پارتیشن ها را در یک DataFrame کاهش می دهد.

Repartition: روش پارتیشن مجدد می تواند برای افزایش یا کاهش تعداد پارتیشن ها در یک DataFrame استفاده شود.

تفاوت بین ادغام و تقسیم مجدد

الگوریتم تقسیم مجدد داده ها را به هم ریخته و پارتیشن هایی با اندازه مساوی از داده ها ایجاد می کند. coalesce پارتیشن های موجود را ترکیب می کند تا از درهم ریختگی کامل جلوگیری کند. از repartition() می توان برای افزایش یا کاهش تعداد پارتیشن ها استفاده کرد، اما شامل جابجایی داده های سنگین در سراسر خوشه است. از طرف دیگر، coalesce() را می توان تنها برای کاهش تعداد پارتیشن ها استفاده کرد. در بیشتر موارد، coalesce() یک shuffle را راه اندازی نمی کند.