پردیس علوم
دانشکده ریاضی، آمار و علوم کامپیوتر

آشنایی با هدوپ

نگارنده

آشنا گرگان محمدی

استاد: دکتر زهرا رضایی قهرودی

پروژه درس پردازش و مدل‌بندی مه‌داده‌ها

آبان ۱۴۰۰

چکیده

ما امروزه در دنیایی از مه‌داده‌ها زندگی می‌کنیم. لذا، ذخیره، پردازش و تحلیل این مه‌داده‌ها از اهمیت بالایی برخوردار است. یکی از محبوب‌ترین پلتفورم‌ها جهت ذخیره و پردازش مه‌داده‌ها، هدوپ است. هدوپ از زمان تولد تا کنون به یکی از بزرگترین اکوسیستم‌های تحلیل و پردازش مه‌داده‌ها بدل شده است؛ چنانکه بخش عمده‌ای از نرم‌افزارهای این حوزه را دربر می‌گیرد. در این گردایه، با هدوپ و تاریخچهٔ آن آشنا می‌شویم و نحوهٔ تعامل با آن را می‌آموزیم.

کلمات کلیدی: مه‌داده، هدوپ، اچ‌دی‌اف‌اس، یارن، پایتون

پیشگفتار

یکی از متداول‌ترین راه‌حل‌ها جهت ذخیره و پردازش مه‌داده‌ها، به‌کارگیری از سیستم‌های توزیع‌شده است. هدوپ بستری برای مدیریت و ذخیرهٔ داده‌ها بر روی یک خوشه از سیستم‌های محاسباتی که بر روی یک شبکه قرار گرفته‌اند، فراهم می‌کند. همچنین، روش‌هایی جهت پردازش کارآمد داده‌ها و مدیریت وظایف و کارها ارائه می‌دهد. هدف این گردایه، آشنا کردن خواننده با مفاهیم اولیهٔ هدوپ و مبانی کار با آن می‌باشد. این گزارش به‌گونه‌ای تنظیم شده است تا نقشهٔ راهی برای شروع کار با هدوپ ارائه دهد و کتابچهٔ راهنمایی برای هر فردی که در ابتدای این راه است، فراهم آورد.

در همین راستا، فصل اول به بیان مفاهیم و تاریخچهٔ هدوپ می‌پردازد. در این فصل، پس از بیان تعریف هدوپ و تاریخچهٔ پیدایش و گسترش آن، به تعریف دقیق‌تری از واحدهای اصلی آن پرداخته می‌شود. در انتها نیز برخی نرم‌افزارها و پروژه‌های مرتبط با اکوسیستم هدوپ را به‌صورت خلاصه معرفی می‌کنیم.

فصل‌های بعد با هدف آموزش کار عملی با هدوپ تهیه و تنظیم شده‌اند. در فصل دوم، ابتدا به چگونگی نصب و راه‌اندازی هدوپ پرداخته می‌شود. سپس، در فصل سوم، نحوهٔ کار با سیستم فایل‌های توزیع‌شدهٔ هدوپ در خط فرمان و در برنامه‌های پایتون بررسی می‌شود. در نهایت، با بررسی مثالی عملی، نحوهٔ پردازش داده‌ها در هدوپ به‌کمک موتور مشهور نگاشت‌کاهش را در زبان پایتون نمایش می‌دهیم.

فصل پنجم نیز به جمع‌بندی مطالب و مرور مفاهیم و کارهای صورت گرفته می‌پردازد. همچنین، منابع بیشتری جهت مطالعهٔ دقیق‌تر به افراد علاقه‌مند پیشنهاد می‌شود.

فهرست مطالب

فصل ۱

مفاهیم اولیه

در عصر مه‌داده‌ها، ذخیره و تحلیل حجم گستردهٔ داده‌ها که روزانه با سرعت و در تنوع بالا در حال تولید هستند، چالشی اساسی است. در راستای پاسخ به نیازهای امروز در حوزهٔ مه‌داده‌ها، بهره‌گیری از ساختارهای توزیعی به عنوان یک راه حل عملی پیشنهاد می‌شود. سیستم‌های گوناگونی نیز در جهت پیاده‌سازی این راه حل توسعه یافته‌اند که هدوپ[1] یکی از آن‌هاست. در این فصل، با هدوپ و تاریخچه آن آشنا خواهیم شد و در ادامه، به بررسی مفاهیم اصلی و پروژه‌های اکوسیستم هدوپ می‌پردازیم.

۱.۱ هدوپ چیست؟

هدوپ یک چارچوب[2] متن‌باز[3] است که امکان خوشه‌بندی[4] سیستم‌های محاسباتی متعدد جهت تحلیل داده‌های در مقیاس بزرگ را به صورت موازی و سریع فراهم می‌کند [۳، ۱۱، ۸، ۶]. به بیان دقیق‌تر، فایل‌ها در هدوپ به بلوک‌های بزرگی تقسیم شده و بر روی گره‌های[5] (سیستم‌های محاسباتی) موجود در خوشه توزیع می‌شوند. سپس، کدها بر روی گره‌ها منتقل می‌شوند و کار پردازش دادگان به صورت موازی در گره‌ها انجام می‌پذیرد. بدین ترتیب، نه تنها محلیت دادگان[6]

[1] Hadoop
[2] Framework
[3] Open-source
[4] Clustering
[5] Node
[6] Data locality

حفظ می‌شود، بلکه پردازش‌ها به مراتب سریع‌تر صورت می‌گیرند [۱۱ ، ۶].
هدوپ شامل چهار واحد می‌باشد که عبارتند از:

۱. سیستم فایل‌های توزیع‌شدهٔ هدوپ (اچ‌دی‌اف‌اس)[۷]: یک سیستم فایل‌های توزیع‌شده است که به سادگی بر روی سیستم‌های استاندارد موجود در بازار قابل اجراست.

۲. نگاشت‌کاهش[۸]: چارچوبی است که امکان انجام محاسبات به صورت موازی بر روی دادگان را فراهم می‌آورد.

۳. یک سیستم مدیریت منابع دیگر (یارن)[۹]: این واحد، گره‌ها و استفادهٔ منابع در خوشه را پایش و مدیریت می‌کند و وظیفهٔ زمان‌بندی[۱۰] کارها[۱۱] و وظایف[۱۲] را بر عهده دارد.

۴. واحد عام هدوپ[۱۳]: این واحد شامل ابزارهای عامی به زبان جاوا است که در تمامی واحد قابل استفاده می‌باشند [۳، ۱۱ ، ۸].

هدوپ عمدتاً به زبان جاوا نوشته شده است و تنها بخشی از کدهای بومی آن به زبان سی و شِل می‌باشند. این امر موجب شده است که برنامه‌های جاوا ساده‌ترین راه حل برای کار با هدوپ باشند. با این حال، رابط‌ها[۱۴] و کتابخانه‌هایی[۱۵] برای مدیریت اچ‌دی‌اف‌اس و نگارش نگاشت‌کاهش در زبان‌های دیگر، از جمله پایتون، نیز فراهم شده‌اند. همچنین، پروژه‌های دیگر در این اکوسیستم قابلیت‌های بیشتری را در راستای سهولت برنامه‌نویسی به زبان‌های دیگر فراهم کرده‌اند [۱].
معماری فیزیکی هدوپ به صورت ارباب‌ـبرده‌ای[۱۶] است؛ بدین معنا که گره‌هایی در خوشه (گره‌های ارباب) مسئول مدیریت گره‌های دیگر (گره‌های برده) هستند. گره‌های ارباب اساساً متشکل از گره نام[۱۷]، گره نام ثانوی[۱۸] و دنبال‌کنندهٔ کار[۱۹]، و گره‌های برده شامل چندین گره

[7] Hadoop Distributed File System (HDFS)
[8] MapReduce
[9] Yet Another Resource Negotiator (YARN)
[10] Scheduling
[11] Job
[12] Task
[13] Hadoop common
[14] Interface
[15] Library
[16] Master-slave
[17] Name node
[18] Secondary name node
[19] Job tracker

داده[۲۰] و دنبال‌کنندهٔ وظایف[۲۱] است (شکل ۱.۱(آ)) [۱۱، ۸].
از دیدگاه عملکردی، می‌توان چهار لایه برای هدوپ مفروض شد:

۱. لایهٔ حافظهٔ توزیع‌شده: این لایه دربرگیرندهٔ اچ‌دی‌اف‌اس است. هر گره در یک خوشهٔ هدوپ منابع و پردازش‌های مختص به خود را دارد. لذا، دادهٔ ورودی به بلوک‌هایی تقسیم می‌شود که به صورت توزیع‌شده و تکرارشونده ذخیره می‌شوند.

۲. لایهٔ مدیریت منابع خوشه: هدوپ باید بتواند هماهنگی خوبی میان گره‌های خوشه ایجاد کند تا کاربرها و برنامه‌ها بتوانند منابع را به‌طور مؤثر به اشتراک بگذارند. به‌طور سنتی، نگاشت‌کاهش مسئول مدیریت منابع و پردازش داده‌ها بود. با معرفی یارن، این دو عملکرد تفکیک شدند و یارن به عنوان یگانه ابزار مدیریت منابع خوشه شناخته شد.

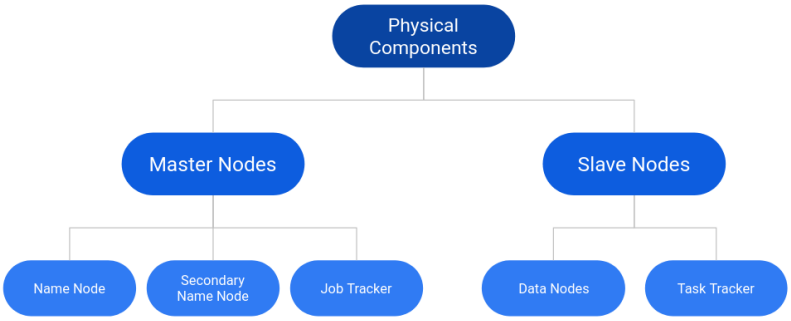
۳. لایهٔ چارچوب‌های پردازشی: این لایه دربرگیرندهٔ برخی چارجوب‌های اکوسیستم هدوپ مانند اسپارک[۲۲]، استورم[۲۳] و تِز[۲۴] و به‌ویژه موتور نگاشت‌کاهش است. این چارچوب‌ها وظیفهٔ تحلیل و پردازش داده را برعهده دارند.

۴. لایهٔ رابط برنامه‌نویسی برنامه[۲۵]: این لایه شامل ابزارها، برنامه‌ها و چارچوب‌های دیگری است که نیازهای متنوع‌تری را برآورده می‌کنند. این ابزارها و برنامه‌ها پروژه‌هایی هستند که در اکوسیستم هدوپ در حال توسعه می‌باشند (شکل ۱.۱(ب)) [۵].

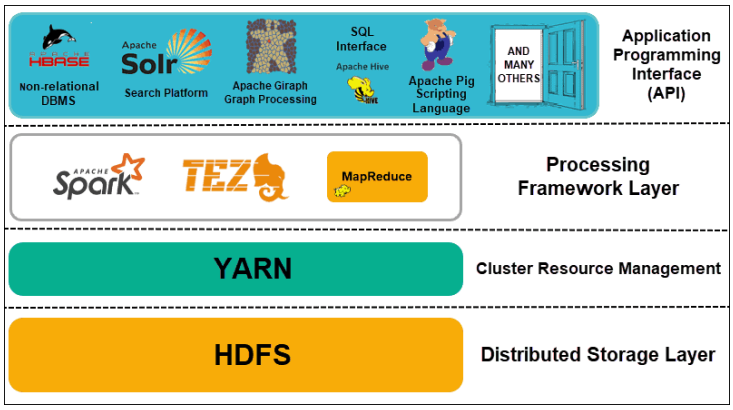
۲.۱ تاریخچه

داگ کاتینگ[۲۶]، خالق پروژهٔ لوسین[۲۷] (یک کتابخانهٔ جستجوی متن)، به همراه مایک کافارِلا[۲۸] در صدد ایجاد یک موتور جستجوی وب[۲۹] متن‌باز بودند. این پروژه که ناچ[۳۰] نام داشت، در

[20] Data node
[21] Task tracker
[22] Spark
[23] Storm
[24] Tez
[25] Application programming interface (API)
[26] Doug Cutting
[27] Lucene
[28] Mike Cafarella
[29] Web search engine
[30] Nutch



(آ)



(ب)

شکل ۱.۱: (آ) معماری فیزیکی هدوپ. (ب) معماری عملکردی هدوپ.

حقیقت بخشی از پروژه لوسین در بنیاد نرم‌افزاری آپاچی بود. کاتینگ و کافارلا می‌دانستند که ساخت و پیاده‌سازی یک موتور جستجو از پایه بسیار هزینه‌بر است. با این حال، این ریسک را پذیرفته و در سال ۲۰۰۲، این پروژه را کلید زدند و سیستم جستجو و خزشگر وب خود را توسعه دادند. از اولین چالش‌هایی که در مقابل آن دو قرار گرفت، حجم زیاد صفحات وب بود. در همین زمان و در سال ۲۰۰۳، گوگل معماری خود در حل این چالش، تحت عنوان سیستم فایل‌های توزیع‌شدهٔ گوگل را طی مقاله‌ای معرفی کرد. خیلی زود معماری مشابهی در ناچ تحت عنوان سیستم فایل‌های توزیع‌شدهٔ ناچ ارائه شد و در سال ۲۰۰۴ به صورت متن‌باز پیاده‌سازی شد. در همین سال، گوگل مقالهٔ دیگری منتشر کرد که در آن موتور نگاشت‌کاهش را به جهانیان معرفی می‌کرد. با افزودن موتور نگاشت‌کاهش به ناچ، در سال ۲۰۰۵ این پروژه به بلوغ نسبی خود رسید.

توسعه‌دهندگان بر این باور بودند که سیستم فایل‌های توزیع‌شدهٔ ناچ و پیاده‌سازی آن‌ها از نگاشت‌کاهش محدود به حوزهٔ جستجو نیست. لذا در سال ۲۰۰۶، از ناچ به سمت یک زیرپروژهٔ دیگری از لوسین تحت عنوان هدوپ مهاجرت کردند و اولین نسخهٔ هدوپ منتشر شد. ملحق شدن کاتینگ به یاهو[۳۱] در همین سال نیز موجب شد تا بستر مناسبی جهت توسعهٔ هدوپ به عنوان یک سیستم قابل اجرا در مقیاس وب فراهم شود. بدین ترتیب، یاهو به یکی از اولین استفاده‌کنندگان هدوپ بدل شد. هدوپ در سال ۲۰۰۸ به یک پروژهٔ مستقل در بنیاد نرم‌افزاری آپاچی تبدیل شد که مسیر موفقیت آن را هموارتر کرد و موجب شد تا از اجتماعی متنوع و فعال برخوردار شود. در این سال، علاوه بر یاهو، شرکت‌هایی نظیر فیس‌بوک[۳۲]، لست‌دات‌اف‌ام[۳۳] و نیویورک تایمز[۳۴] از این تکنولوژی بهره می‌بردند.

هدوپ نام خود را وام‌دار فرزند کاتینگ است! در زمانی که کاتینگ به‌دنبال نامی برای پروژه بود، فرزند او در حال بازی با عروسک فیل زرد خود به نام هدوپ بود. کاتینگ در این رابطه می‌گوید:

> [هدوپ] نامی کوتاه، به‌راحتی قابل تلفظ و بی‌معنی بود که هیچ جای دیگر به‌کار نرفته بود. این موارد معیارهای من در نامگذاری هستند. کودکان در تولید چنین نام‌هایی تبحر دارند.

و اینگونه شد که هدوپ با نماد فیل زرد برای این پروژه انتخاب شد.

همانطور که پیش‌تر اشاره شد، نسخهٔ اولیهٔ هدوپ در سال ۲۰۰۶ منتشر شد. این پروژه در سال ۲۰۱۱ به‌طور رسمی به صورت متن‌باز در دسترس قرار گرفت. در این سال‌ها، نسخه‌های اصلاحی دیگری از پروژه نیز منتشر شد که تغییراتی جزئی نسبت به نسخه اولیه داشتند. بعد از حدود ۸ سال، در سال ۲۰۱۴ نسخهٔ دوم با بهره‌گیری از یارن، که در سال ۲۰۱۲ معرفی شده بود، ارائه شد [۱۱]. اصلاحات متعددی بر روی این نسخه نیز صورت گرفت تا نهایتاً در سال ۲۰۲۰ نسخهٔ سوم آن با پشتیبانی از نسخهٔ هشتم جاوا منتشر شد (توسعهٔ نسخه هفتم جاوا در سال ۲۰۱۵ متوقف شد). نسخهٔ سوم بهبودهایی در راستای مقاومت در برابر خرابی‌های سیستم و کاهش سربار حافظه اعمال کرده است. همچنین، امکان بهره‌گیری از کارت گرافیک را در خوشه‌ها فراهم آورده است [۲، ۴].

---

[31] Yahoo!
[32] Facebook
[33] Last.fm
[34] New York Times

۳.۱ اچ‌دی‌اف‌اس

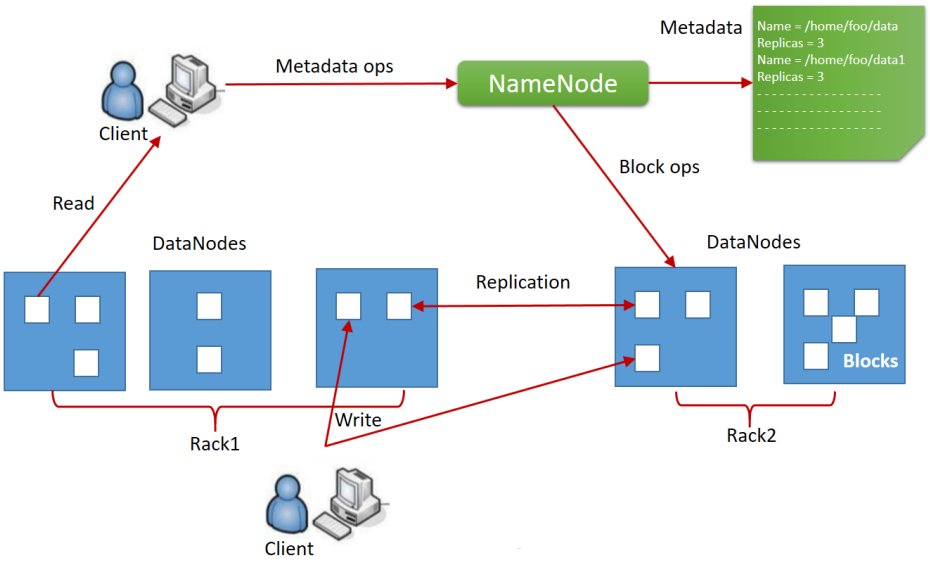
همان‌طور که پیش‌تر اشاره شد، اچ‌دی‌اف‌اس یا همان سیستم فایل‌های توزیع‌شدهٔ هدوپ، به‌منظور ذخیرهٔ دادگان حجیم به صورت توزیع‌شده بر روی سیستم‌های موجود در یک خوشهٔ هدوپ و با الگوی دسترسی به صورت جریانی از داده‌ها به‌کار می‌رود. در این تعریف، منظور از الگوی دسترسی به صورت جریانی از داده‌ها آن است که منطق طراحی اچ‌دی‌اف‌اس بر پایهٔ الگوی «یک بار نوشتن و چندین بار خواندن» است. به عبارت دیگر، اچ‌دی‌اف‌اس به‌منظور انجام به‌روزرسانی‌های متعدد بر روی داده‌ها ساخته نشده است، بلکه در راستای تحلیل گسترده و چندباره به‌وجود آمده است.

اچ‌دی‌اف‌اس به‌گونه‌ای طراحی شده است که از قابلیت تحمل خطا[۳۵] و دسترس‌پذیری[۳۶] بالایی برخوردار است. در این طراحی، داده به بلوک‌های بزرگی (معمولاً ۱۲۸ یا ۲۵۶ مگابایت) تقسیم شده و کپی‌های مجزایی از آن (سه نسخه به‌صورت پیش‌فرض) در گره‌های متنوع در رک‌های[۳۷] متعدد ذخیره می‌شوند (منظور از رک، مجموعه‌ای فیزیکی از چندین گره در خوشه است). بدین ترتیب، اگر یکی از گره‌ها و یا حتی یک رک دچار اختلال شود، تأثیر آن در کل سیستم قابل چشم‌پوشی است و عملیات پردازش داده‌ها بدون هیچ مشکلی ادامه می‌یابد. الگوریتم تکرار بلوک داده در گره‌های رک‌های مختلف که به الگوریتم رک‌ـآگاه[۳۸] معروف است، بدین صورت عمل می‌کند که تکرار اول را در همان رک کارخواه[۳۹] قرار می‌دهد. سپس، یک رک دیگر به‌صورت تصادفی انتخاب شده، و دو تکرار روی گره‌های مجزا در آن رک قرار می‌گیرند. در جهت جایدهی تکرارهای دیگر از بلوک داده در خوشه، گره‌های مجزا به صورت تصادفی از مجموعهٔ تمام گره‌های موجود در خوشه (بدون توجه به رک) انتخاب شده و عملیات ذخیره‌سازی انجام می‌شود.

اچ‌دی‌اف‌اس شامل گره نام، گره نام ثانوی و چندین گره داده است. پس از شکستن داده به تعدادی بلوک، این بلوک‌ها به‌کمک گره‌های داده در سرورهای برده ذخیره می‌شوند. ابرداده‌های[۴۰] این بلوک‌ها، شامل نام فایل، مجوزهای فایل، شناسه‌ها، مکان ذخیره و تعداد تکرارها[۴۱]، در حافظهٔ داخلی گره نام ذخیره می‌شوند (شکل ۲.۱). آنچه از این توضیحات برمی‌آید آن است که اگر گره نام دچار اختلال شود، اچ‌دی‌اف‌اس دیگر قادر نخواهد بود تا دادگان را بر روی گره‌های داده توزیع کند. بنابراین، اختلال در گره نام موجب اختلال در کل خوشه می‌شود. برای حل این مشکل، از

[35] Fault-tolerance
[36] Availability
[37] Rack
[38] Rack-aware algorithm
[39] Client
[40] Metadata
[41] Replica

گره نام ثانوی بهره گرفته می‌شود. در نسخه‌های اولیهٔ هدوپ، گره نام ثانوی نسخهٔ پشتیبان[۴۲] اولیه از گره نام بود. گره نام ثانوی به‌طور معمول یک کپی از ابرداده‌های فعلی و پیشینهٔ ویرایش‌های[۴۳] صورت‌گرفته از گره نام تهیه و آن‌ها را ادغام می‌کند.



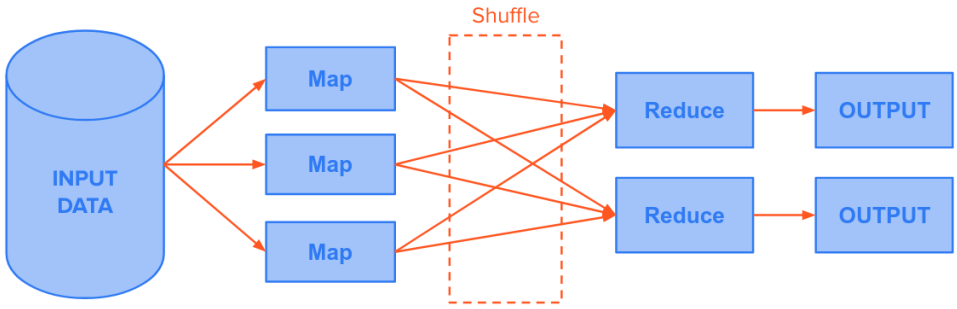
شکل ۲.۱: شمای کلی سیستم فایل‌های توزیع‌شدهٔ هدوپ.

فرآیند رفع مشکل در صورت بروز اختلال در گره نام به کمک پشتیبان موجود در گره نام ثانوی، نیازمند دخالت سرپرست سیستم است. در راستای اتوماسیون فرآیند بازیابی حالت عادی پس از ایجاد اختلال در گره نام، در نسخهٔ دوم هدوپ جفتی از گره‌های نام با وضعیت‌های فعال–آماده‌به‌کار[۴۴] لحاظ شده‌اند. بدین ترتیب، در صورت بروز مشکل در گره نام فعال، گره نام آماده‌به‌کار وظایف را از سر می‌گیرد و وضعیت آن به فعال تغییر می‌کند. لازم به ذکر است که در راستای افزایش دسترس‌پذیری، دو گره نام از یک حافظهٔ اشتراکی با دسترس‌پذیری بالا برای ذخیرهٔ پیشینهٔ ویرایش‌ها بهره می‌برند تا تغییر گره نام فعال در کمترین زمان ممکن رخ دهد (برای اطلاعات بیشتر در خصوص چگونگی انتخاب این حافظهٔ اشتراکی به کلمات کلیدی NFS filer و Quorum journal manager رجوع کنید) [۱۱، ۹، ۱، ۵].

[42] Backup
[43] Edit logs
[44] Active-standby

۴.۱ موتور نگاشت‌کاهش

همان‌طور که پیش‌تر اشاره شد، موتور نگاشت‌کاهش اولین بار در مقاله‌ای توسط گوگل مطرح شد. موتور نگاشت‌کاهش به‌منظور پردازش سریع و موازی داده‌های توزیع‌شده در یک خوشه به‌کار می‌رود و کارهای مرتبط با پردازش داده را به دو فاز نگاشت و کاهش تقسیم می‌کند. در حالت کلی، می‌توان گفت که تابع نگاشت عمدتاً به آماده‌سازی داده پرداخته، نتایجی به صورت زوج‌های کلید_مقدار[۴۵] تولید می‌کند که پس از درهم‌آمیزی[۴۶] و مرتب‌سازی، به عنوان ورودی به تابع کاهش، که محاسبات و تجمیع نتایج بر روی داده‌ها را انجام می‌دهد، داده می‌شوند. شکل ۳.۱ شمای کلی موتور نگاشت‌کاهش را نشان می‌دهد.



شکل ۳.۱: شمای کلی موتور نگاشت‌کاهش.

هر کار نگاشت‌کاهش، یک واحد از عملیات مورد نظرِ کارخواه است و متشکل از داده‌های ورودی، برنامهٔ نگاشت‌کاهش و اطلاعات پیکربندی می‌باشد. در هدوپ، این کارها به دو وظیفهٔ نگاشت و کاهش تقسیم می‌شوند. وظایف به‌کمک یارن (که در بخش بعد بیشتر به آن خواهیم پرداخت) زمان‌بندی و بر روی گره‌های خوشه اجرا می‌شوند. در صورت بروز هرگونه مشکل در یکی از این وظایف، عملیات مجدد زمان‌بندی می‌شود تا بر روی گره دیگری اجرا شود. علاوه‌براین، هدوپ دادهٔ ورودی را به قطعه‌های با طول برابر تقسیم می‌کند. سپس، هدوپ یک وظیفهٔ نگاشت به‌ازای هر قطعه ایجاد می‌کند که در حقیقت، تابع نگاشتی که کاربر تعریف کرده است را بر روی رکوردهای آن قطعه اجرا می‌کند. هدوپ تلاش می‌کند تا انجام وظیفهٔ نگاشت را به گره‌ای محول کند که داده در آن ذخیره شده است. با این حال، در شرایطی ممکن است آن گره در حال انجام وظیفهٔ دیگری باشد. در این صورت، ابتدا سعی می‌کند گره‌ای بیکار در همان رک بیابد و اگر موفق

[45] Key-value
[46] Shuffle

نشود، از گره‌ای خارج از رک استفاده خواهد کرد. با توجه به آنچه گفته شد، اندازهٔ بهینه برای قطعات ورودی به اندازهٔ سایز بلوک‌های اچ‌دی‌اف‌اس است؛ زیرا در این صورت، تضمین می‌شود که هر وظیفهٔ نگاشت قابل اجرا بر روی یک گره داده خواهد بود و در صورت اختلال در یک گره داده، عملیات نگاشت بر روی دیگر گره‌ها بدون مشکل و به‌صورت مستقل قابل انجام خواهد بود.

تابع نگاشت به‌ازای اجرا بر روی قطعات ورودی، زوج‌هایی به‌صورت کلید_مقدار تولید می‌کند. این زوج‌ها داده‌هایی میانی هستند که در ادامه به عنوان ورودی به تابع(های) کاهش داده خواهند شد و استفادهٔ دیگری ندارند. لذا، این زوج‌ها بر روی اچ‌دی‌اف‌اس ذخیره نمی‌شوند و تنها بر روی حافظهٔ داخلی نوشته می‌شوند. پیش از آن که این زوج‌ها به تابع کاهش وارد شوند، درهم آمیخته شده و با توجه به کلید مرتب می‌شوند. در نهایت، تابع کاهش زوج‌ها را به‌ترتیب خوانده و پس از تجمیع و پردازش، خروجی را برای ذخیره بر روی اچ‌دی‌اف‌اس تولید می‌کند. در صورت وجود حجم بالای دادگان، اگر تجمیع و پردازش مورد نظر قابل شکستن باشد، یک تابع تلفیق‌گر[۴۷] نیز تعریف می‌شود. تابع تلفیق‌گر به صورت مستقل بر روی خروجی‌های هر تابع نگاشت اجرا می‌شود و بار محاسباتی درهم آمیختن و مرتب‌سازی خروجی‌های تابع نگاشت را کاهش می‌دهد.

برای مثال، فرض کنید داده‌های مربوط به دسترسی افراد به یک وبسایت در اختیار است و می‌خواهیم تعداد درخواست برای دسترسی هر فایل را محاسبه کنیم. بدین منظور، تابع نگاشتی نوشته‌ایم که به‌ازای هر وقوع نام یک فایل در قطعهٔ ورودی، نام فایل را به همراه عدد ۱ در خروجی می‌نویسد. بدون نوشتن تابع تلفیق‌گر، لازم است تمامی این زوج‌های (نام فایل،۱) به ترتیب نام فایل مرتب شوند و سپس به کمک تابع کاهش و با تجمیع ۱ها، تعداد کل درخواست‌ها برای آن فایل محاسبه شود. در راستای کاهش محاسبات جهت مرتب‌سازی خروجی‌های تابع نگاشت، می‌توان یک تابع تلفیق‌گر نوشت که خروجی هر قطعه پس از نگاشت را مرتب و تعداد وقوع هر نام فایل را بشمارد و در خروجی بنویسد و سپس، خروجی آن مرتب و با تابع مشابهی، عملیات کاهش صورت گیرد. کد مربوط به این مثال را به صورت صریح در بخش‌های آینده مشاهده خواهیم کرد [۱۱، ۹، ۶].

۵.۱ یارن

در نسخهٔ اول هدوپ، موتور نگاشت‌کاهش به تنهایی بار پردازش داده و مدیریت منابع را به دوش می‌کشید. به مرور زمان، اهمیت تفکیک این دو عملیات پررنگ شد و به توسعهٔ یارن ختم شد. یارن ابزار مدیریت خوشهٔ پیش‌فرض در نسخه‌های دوم و سوم هدوپ است که امکان تعریف رابط‌ها

[47] Combiner

جهت پیاده‌سازی موتور پردازشی سفارشی‌سازی شده را نیز می‌دهد. یارن شامل دو دیمن[۴۸] است: یک مدیر منابع که مصرف منابع در خوشه را مدیریت می‌کند و تعدادی مدیر گره که روی هر گره درون خوشه قرار دارند و کانتینرها[۴۹] را پایش و آغاز می‌کنند. هر کانتینر یک فرآیند مختص برنامه را با توجه به محموعه‌ای از محدودیت‌ها در منابع، اجرا می‌کند.

به‌منظور اجرای یک برنامه بر روی یارن، کارخواه یک درخواست به مدیر منابع ارسال می‌کند. مدیر منابع فرمانی به مدیر گره جهت آغاز یک اربابِ برنامه[۵۰] صادر می‌کند که در یک کانتینر قرار می‌گیرد. سپس، ارباب برنامه خود را به مدیر منابع معرفی می‌کند تا اجازهٔ ارتباط با گره(های) نام، یافتن محل ذخیرهٔ بلوک‌های داده و انجام محاسبات لازم در وظایف نگاشت و کاهش برای پردازش داده را اخذ کند. پس از آن، ارباب برنامه منابع مورد نیاز را از مدیر منابع درخواست می‌کند و به تعامل با منابع مورد نیاز در طول چرخهٔ حیات کانتینر، ادامه می‌دهد. مدیر منابع نیز به زمان‌بندی منابع درخواستی ارباب برنامه می‌پردازد و صفی از درخواست‌ها تشکیل می‌دهد. با آزاد شدن منابع، آن‌ها را در اختیار ارباب برنامه جهت تخصیص به گره بردهٔ مشخص، قرار می‌دهد. ارباب برنامه درخواستی به مدیر آن گره برده ارسال می‌کند تا یک کانتینر حاوی متغیرها، توکن‌های احراز هویت و رشتهٔ دستورات فرآیند مورد نظر ایجاد کند. در تمام لحظات، ارباب برنامه بر اجرای فرآیندها نظارت می‌کند و در صورت بروز هرگونه اختلال، فرآیند را مجدد راه‌اندازی می‌کند. با پایان وظایف، ارباب برنامه نتایج را به برنامهٔ کارخواه ارسال می‌کند و پس از اعلام سیگنال خاتمه به مدیر منابع، اجرای خود را نیز متوقف می‌کند [۱۱، ۱، ۵].

زمان‌بندی بر روی یک خوشه از اهمیت بالایی برخوردار است. سه زمان‌بندی که یارن از آن‌ها بهره می‌برند، عبارتند از:

۱. فیفو[۵۱]: این زمان‌بند برنامه‌ها را در یک صف قرار می‌دهد و به ترتیب ارسال، آن‌ها را اجرا می‌کند؛ بدین نحو که ابتدا منابع به درخواست‌های برنامهٔ اول در صف اختصاص می‌یابند و پس از پایان انجام وظایف مربوط به درخواست‌ها، منابع در اختیار برنامهٔ بعدی در صف قرار می‌گیرند.

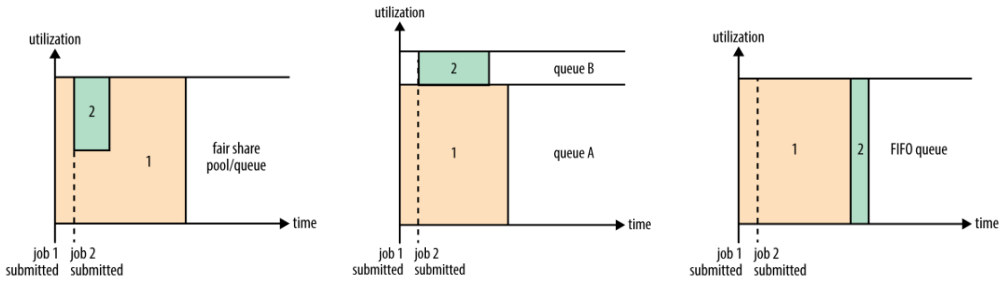
۲. زمان‌بند ظرفیت[۵۲]: در این زمان‌بند، یک صف اختصاصی مجزا وجود دارد که امکان اجرای کارهای کوچک را به محض ارسال فراهم می‌کند. بهره‌وری این زمان‌بند نسبت به زمان‌بند

[48] Daemon
[49] Container
[50] Application master
[51] FIFO (First in, first out)
[52] Capacity Scheduler

فیفو کمتر است؛ چراکه بخشی از ظرفیت صف برنامه‌ها برای کارهای این صفِ اختصاصی محفوظ می‌ماند و برنامه‌های بزرگ به‌طور معمول دیرتر از حالتی که از زمان‌بند فیفو استفاده شده باشد، خاتمه می‌یابند.

۳. زمان‌بند منصفانه[۵۳]: در این زمان‌بند، منابع به صورت پویا به کارهای در حال اجرا تخصیص می‌یابند و ظرفیت محفوظی وجود ندارد [۱۱].

شکل ۴.۱ تفاوت بین این سه زمان‌بند را به‌خوبی نشان می‌دهد.



(ج) (ب) (آ)

شکل ۴.۱: (آ) زمان‌بند فیفو. (ب) زمان‌بند ظرفیت. (ج) زمان‌بند منصفانه.

۶.۱ اکوسیستم هدوپ

هدوپ به این واحدهای اصلی محدود نمی‌شود، بلکه بسیاری دیگر از پروژه‌ها و برنامه‌های وابسته را نیز شامل می‌شود. در این بخش به بررسی اجمالی برنامه‌های موجود در اکوسیستم هدوپ می‌پردازیم.

کاساندرا: کاساندرا[۵۴] یک پایگاه دادهٔ متن‌باز و توزیع‌شده با مدل ذخیرهٔ ستونی نواس‌کیوال[۵۵] است. این پایگاه داده نقطهٔ شکست واحد ندارد، گسترش‌پذیر است و زبان پرس‌وجوی[۵۶] اختصاصی و مشابه با زبان اس‌کیوال را دارد.

اچ‌بیس: اچ‌بیس[۵۷] یک پایگاه دادهٔ توزیع‌شدهٔ ستون‌محور است که بر روی اچ‌دی‌اف‌اس ساخته

[53] Fair scheduler
[54] Cassandra
[55] NoSQL
[56] Query
[57] HBase

شده است. اچبیس برنامه‌ای در اکوسیستم هدوپ است که جهت رفع نیاز خواندن و نوشتن بلادرنگ در مجموعهٔ دادگان خیلی بزرگ توسعه یافته است.

اوزون: اوزون[۵۸] یک پایگاه توزیع‌شده برای ذخیرهٔ اشیاء به‌صورت کلید_مقدار است که می‌تواند فایل‌های بزرگ و کوچک را به‌طور مشابه و کارا مدیریت کند. اوزون برنامه‌ای است که اخیراً معرفی شده است و همچنان در حال توسعه است. این برنامه می‌تواند به‌راحتی در کنار دیگر برنامه‌های اکوسیستم هدوپ، از جمله اسپارک، کار کند.

آورو: آورو[۵۹] یک سیستم مسلسل‌سازی[۶۰] داده است. آورو از جیسون[۶۱] برای تعریف انواع داده و پروتکل‌های داده بهره می‌برد و داده را به‌صورت فشرده و دودویی مسلسل‌سازی می‌کند.

زوکیپر: زوکیپر[۶۲] سرویس هماهنگ‌سازی سیستم‌های توزیع‌شده در هدوپ است. زوکیپر بستری برای ذخیرهٔ سلسله‌مراتبی کلید_مقدارها ارائه می‌کند که به‌منظور فراهم آوردن پیکربندی و همزمانی به‌صورت توزیع‌شده توسعه یافته است.

هایو: هایو[۶۳] یک پروژهٔ نرم‌افزاری انبار داده، توسعه یافته توسط فیس‌بوک است که امکان تحلیل و پرس‌وجو را در هدوپ فراهم می‌کند. هایو نیز یک رابط زبانی مشابه اس‌کیوال جهت انجام پرس‌وجوها ارائه می‌کند.

پیگ: پیگ[۶۴] با افزودن یک انتزاع به موتور نگاشت‌کاهش، موجب تسهیل پردازش داده‌ها می‌شود و برنامه‌نویسی سطح بالا برای نگارش موتور نگاشت‌کاهش فراهم می‌کند.

اسپارک: اسپارک[۶۵] یکی از معروف‌ترین موتورهای تحلیل داده در پردازش داده‌های حجیم است که توسط دانشگاه برکلی توسعه یافته است. اسپارک یک رابط به‌منظور برنامه‌نویسی کل خوشه با موازی‌سازی و تحمل خطای صریح را در اختیار قرار می‌دهد. این موتور سریع متن‌باز امکان پرس‌وجو، پردازش دسته‌ای، تحلیل جریان داده‌ها، پیاده‌سازی و اجرای الگوریتم‌های یادگیری ماشین و نیز تعریف پایگاه داده‌های گراف را فراهم می‌کند. اسپارک از موتور

[58] Ozone
[59] Avro
[60] Serialization
[61] JSON
[62] ZooKeeper
[63] Hive
[64] Pig
[65] https://spark.apache.org/

نگاشت‌کاهش استفاده نمی‌کند و به توانایی آن در نگه داشتن مجموعهٔ دادگان در حافظه بین کارها شهرت دارد.

استورم: استورم[۶۶] یک چارچوب محاسباتی برای پردازش جریان داده‌های توزیع‌شده است. ساختار آن به‌صورت گراف جهت‌دار بدون دور است و عمدتاً به زبان کلوژر[۶۷] نوشته شده است.

تز: تز[۶۸] نیز یک چارچوب پردازشی با ساختار گراف جهت‌دار بدون دور است. تز چارچوبی مبتنی بر یارن است که به‌منظور توسعهٔ برنامه‌های پردازش داده به‌صورت دسته‌ای و تعاملی و طراحی نحوهٔ جریان داده پدید آمده است.

ماهوت: ماهوت[۶۹] یک پروژه جهت فراهم آوردن پیاده‌سازی‌هایی توزیع‌شده و گسترش‌پذیر از الگوریتم‌های یادگیری ماشین و بر پایهٔ جبر خطی است. ماهوت پیش‌تر بر پایهٔ پلتفورم هدوپ نوشته شده بود ولی نسخه‌های جدید آن بر اسپارک تمرکز دارند.

ساب‌مارین: ساب‌مارین[۷۰] یک پلتفورم هوش مصنوعی است که امکان اجرای الگوریتم‌های یادگیری ماشین و یادگیری عمیق را در خوشه‌های توزیع‌شده فراهم می‌کند.

امباری: امباری[۷۱] یک ابزار مبتنی بر وب است که جهت پایش و مدیریت خوشه‌های هدوپ و برنامه‌های مرتبط توسعه یافته است. امباری یک داشبورد مدیریتی را در اختیار کاربر قرار می‌دهد که به‌کمک آن می‌توان سلامت خوشه‌ها را بررسی کرد.

حال که با مفاهیم اولیه و برخی پروژه‌های مرتبط با هدوپ و اهداف آن‌ها آشنا شده‌ایم، می‌توانیم در راستای یادگیری نحوهٔ کار با هدوپ گام برداریم. لذا در ادامه، به بررسی نحوهٔ نصب و راه‌اندازی هدوپ پرداخته و سپس، کار با اچ‌دی‌اف‌اس و نگارش موتور نگاشت‌کاهش، در کنار چالش‌های موجود را به‌صورت عملی و در بستر خط فرمان و برنامه‌های به زبان پایتون مورد مطالعه قرار می‌دهیم.

[66] http://storm.apache.org/
[67] Clojure
[68] https://tez.apache.org/
[69] Mahout
[70] Submarine
[71] Ambari

فصل ۲

نصب و راه‌اندازی هدوپ

۱.۲ حالت‌های نصب هدوپ

هدوپ در سه حالت قابل نصب است:

۱. حالت مستقل[1]: با وجود اینکه هدوپ یک پلتفورم توزیع‌شده است، می‌توان آن را بر روی یک گره و به صورت مستقل نیز نصب کرد. در این حالت، هدوپ مانند یک سیستم که بر روی محیط جاوا اجرا می‌شود، عمل می‌کند. حالت مستقل عمدتاً به هدف عیب‌یابی از برنامه‌های هدوپ مورد استفاده قرار می‌گیرد؛ به‌طور مثال، می‌توانیم صحت برنامه‌های نگاشت‌کاهش را پیش از اجرای آن‌ها در یک خوشه مورد بررسی قرار دهیم.

۲. حال شبه‌توزیع‌شده[2]: در این حالت نیز نصب بر روی یک گره صورت می‌گیرد؛ با این تفاوت که یک شبیه‌سازی از یک خوشهٔ هدوپ بر روی آن گره در اختیار خواهد بود. بنابراین، دیمن‌های مختلف، از جمله گره نام، گره‌های داده، دنبال‌کنندهٔ وظیفه و دنبال‌کنندهٔ کار، همگی بر روی یک ماشین جاوا اجرا می‌شوند.

۳. حال تماماً توزیع‌شده[3]: در این حالت، تعدادی گره در اختیار است که یک خوشهٔ هدوپ را تشکیل می‌دهند. در این تنظیمات، گره نام، گره نام ثانوی و دنبال‌کننده‌های کار بر روی

[1] Standalone mode
[2] Pseudo-distributed mode
[3] Fully-distributed mode

گره ارباب عمل تنظیم می‌شوند و گره‌های داده و دنبال‌کنندهٔ وظایف بر روی گره‌های برده قرار می‌گیرند [۱۱].

در این فصل نحوهٔ نصب آخرین نسخهٔ هدوپ در هر سه حالت را بر روی سیستم‌های مبتنی بر لینوکس‌ـدبیان و ویندوز شرح می‌دهیم. توجه کنید که با توجه به محدودیت‌های موجود، تنها نصب بر روی سیستم لینوکس آزمایش شده است و نصب حالت تماماً‌توزیع‌شده نیز تست نشده است. با این حال، با بررسی منابع مختلف و صحت‌سنجی مراحل گفته شده با توجه به زیرساخت‌های لازم، مطالب به‌منظور نصب و راه‌اندازی هدوپ بر روی سیستم‌های مبتنی بر ویندوز و نیز نصب در حالت تماماً‌توزیع‌شده، به‌صورت تئوری تشریح شده‌اند. مباحث بر پایهٔ منابع [۱۱]، [۱۰] و [۷] و با توجه به تجربهٔ عملی تنظیم شده‌اند.

۲.۲ نصب نیازمندی‌ها

۱.۲.۲ لینوکس‌ـدبیان

پیش از هر چیز، لازم است نرم‌افزارهای سیستم را به‌روز کنیم. بنابراین، ابتدا دستورات زیر را در ترمینال (خط فرمان) اجرا می‌کنیم:

```
$ sudo apt update
$ sudo apt upgrade
```

همان‌طور که در فصل اول اشاره شد، عمدهٔ پروژه هدوپ به زبان جاوا نوشته شده است و سرویس‌های آن نیازمند محیط زمان اجرای جاوا[۴] (جِی‌آرای) و کیت توسعهٔ جاوا[۵] (جِی‌دی‌کِی) است. پکیج اُپن‌جی‌دی‌کی[۶] در سیستم‌های مبتنی بر دبیان لینوکس از جمله اوبونتو هر دو نیازمندی را دربردارد. نسخهٔ سوم هدوپ از نسخهٔ هشتم جاوا پشتیبانی می‌کند و لذا کافیست دستورات زیر را اجرا کنیم:

```
$ sudo apt install openjdk-8-jdk -y
```

پس از پایان یافتن نصب، برای اطمینان از نصب صحیح، می‌توانید نسخهٔ جاوا و کامپایلر آن را به‌کمک دستورات زیر بررسی کنید:

[4] Java runtime environment
[5] Java development kit
[6] OpenJDK

```
$ java -version
openjdk version "11.0.11" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2
    .20.04)
OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.20.04,
     mixed mode, sharing)
$ javac -version
javac 1.8.0_292
```

به منظور اطمینان از عملکرد امن هدوپ، لازم است یک اتصال اس‌اس‌اچ[7] بدون نیاز به پسورد به میزبان محلی[8] برقرار کنیم. به‌منظور نصب نسخه‌های مرتبط با میزبان و کارخواه اس‌اس‌اچ، از دستور زیر استفاده می‌کنیم:

```
$ sudo apt install openssh-server openssh-client -y
```

توصیه می‌شود که یک کاربر بدون دسترسی روت ایجاد شود تا از اشتباهاتی که منجر به تغییر تنظیمات اصلی سیستم می‌شود، جلوگیری شود. دستورات زیر، یک کاربر با نام bigdata و بدون دسترسی روت ایجاد کرده، به محیط آن کاربر وارد شده، و تنظیمات اس‌اس‌اچ را برای اتصال این کاربر به میزبان محلی انجام می‌دهد:

```
$ sudo adduser bigdata
$ su - bigdata
$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
$ ssh localhost
```

۲.۲.۲ ویندوز

برای نصب در ویندوز، همچنان نصب نسخهٔ هشتم جاوا نیاز است. با این حال، جهت فراهم کردن فضای خط فرمان مناسب، ابتدا باید دو نرم افزار پاورشل[9] و یکی از گیت‌بَش[10] یا سون‌زیپ[11]

[7] SSH (Secure shell)
[8] Localhost
[9] PowerShell
[10] Git Bash
[11] 7 Zip

نصب شوند. با توجه به معماری سیستم خود، می‌توانید فایل زیپ جهت نصب پاورشل را از طریق یکی از لینک‌های زیر دریافت کنید.

- معماری ۶۴ بیتی
- معماری ۳۲ بیتی
- معماری اِی‌آرام[۱۲] ۶۴ بیتی
- معماری اِی‌آرام ۳۲ بیتی

به‌علاوه، می‌توانید در خط فرمان سی‌ام‌دی[۱۳] دستور زیر را اجرا کنید:

```
winget search Microsoft.PowerShell
```

سون‌زیپ یا گیت‌بش در فرآیند نصب تنها جهت خارج کردن فایل‌های دودویی هدوپ از حالت فشرده استفاده خواهند شد. با این حال، از آنجایی که گیت‌بش بستر مناسبی برای کار با گیت را نیز فراهم می‌کند، نصب آن توصیه می‌شود. برای نصب این دو نرم‌افزار نیز می‌توانید از مسیرهای زیر اقدام کنید:

- لینک کمکی برای نصب سون‌زیپ (با توجه به معماری سیستم خود، فایل exe مناسب را دانلود کنید)
- لینک نصب گیت‌بش

برای نصب جاوا، پس از دانلود فایل متناسب با معماری سیستم خود از سایت اُراکِل[۱۴]، فایل exe دانلود شده را اجرا کرده و به‌کمک راهنمای تصویری فراهم شده در این سایت، به‌راحتی آن را نصب کنید.

۳.۲ نصب هدوپ

۱.۳.۲ دانلود هدوپ و تنظیم متغیرهای محیط

برای نصب هدوپ، ابتدا به سایت رسمی پروژه آپاچی هدوپ رفته و با انتخاب گزینهٔ binary در مقابل نسخهٔ مورد نظر (شکل ۱.۲(آ))، روی لینکی که جهت دانلود به شما پیشنهاد می‌شود

[12] ARM
[13] cmd
[14] Oracle

(شکل ۱.۲ (ب)) کلیک کنید و یا آن را کپی کنید تا از طریق خط فرمان و به کمک دستورات زیر فایل را دانلود کنید (اگر بر روی سیستم لینوکس کار می‌کنید، ابتدا مطمئن شوید که به عنوان کاربری که در بالا ایجاد کرده‌اید، وارد شوید.):

```
# Ubuntu/Debian
$ wget <mirror-link>
# Windows (PowerShell)
$ client = new-object System.Net.WebClient
$ client.DownloadFile(<mirror-link>, path/to/save/file)
```

پس از اتمام، به محلی که فایل دانلود شده (که یک فایل فشرده با فرمت است) در آن ذخیره شده است رفته و آن را از حالت فشرده خارج نمایید:

```
# Ubuntu/Debian, Windows (Git Bash)
$ cd path/to/file
$ tar xzf file.tar.gz
```

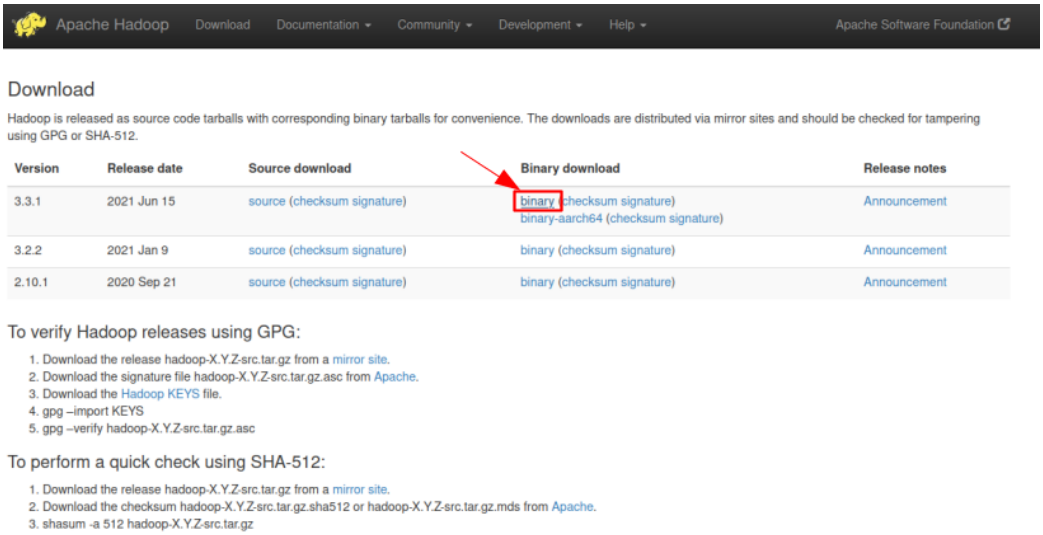
برای نصب هدوپ بر روی ویندوز، لازم است فایل‌های مربوط به ورودی/خروجی بومی هدوپ[15] نیز نصب شوند. فایل‌های مورد نیاز به‌تفکیک نسخهٔ هدوپ در این ریپازیتوری گیتهاب قرار گرفته‌اند و با انتخاب نسخهٔ مورد نظر، تمام فایل‌ها جهت دانلود به شما نمایش داده می‌شوند. فایل‌ها را دانلود کرده و در فولدر `bin` در محل نصب هدوپ قرار دهید.

> توجه! تا این لحظه، تنها فایل‌های ورودی/خروجی بومی هدوپ تا نسخهٔ ۳.۲.۲ در گیت‌هاب قرار داده شده‌اند. لذا هنگام انتخاب نسخهٔ هدوپ جهت نصب بر روی ویندوز، این نکته را در نظر داشته باشید.

مراحل بالا در صورتی‌که دقیق مطابق با دستورالعمل‌ها انجام شوند، بدون خطا خواهند بود و آمادهٔ تعریف تنظیمات لازم برای نصب هدوپ خواهیم بود. پیش از هر چیز، لازم است متغیرهای محیط تنظیم شوند. بدین منظور، اگر از سیستم عامل‌های لینوکس مبتنی بر دبیان استفاده می‌کنید، فایل `~/.bashrc` که شامل تنظیمات محیط است را به کمک یک ویرایشگر متن مانند `nano` باز کرده و عبارت‌های زیر را در انتهای آن اضافه کنید:

```
export HADOOP_HOME=path/to/hadoop-folder
export HADOOP_INSTALL=$HADOOP_HOME
export HADOOP_MAPRED_HOME=$HADOOP_HOME
```

[15] Hadoop native I/O



(آ)

(ب)

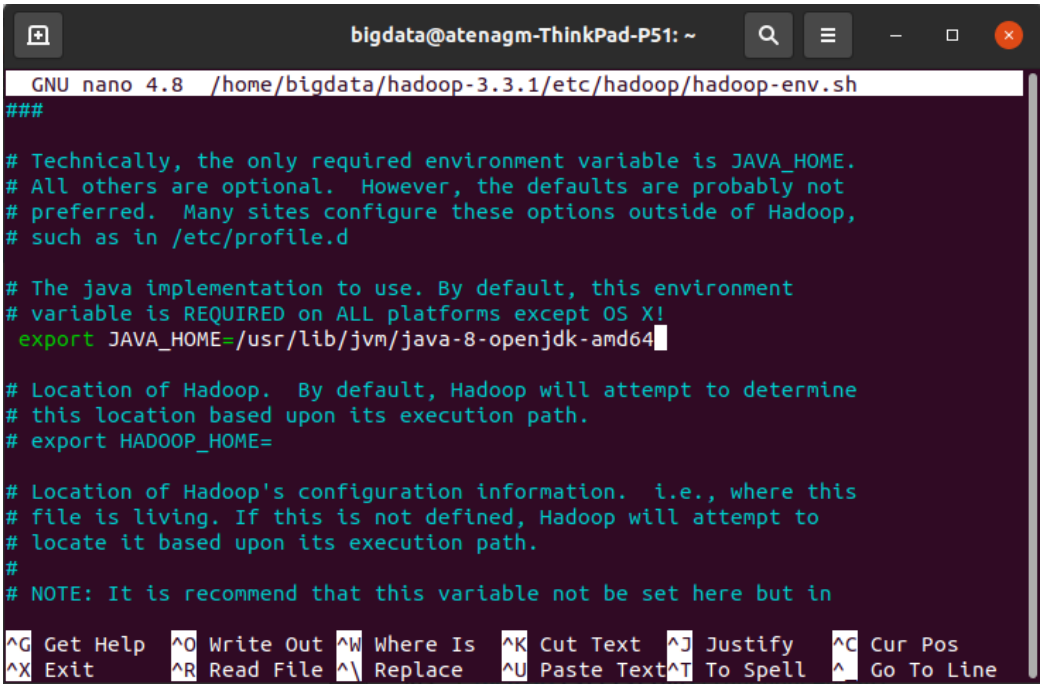
شکل ۱.۲: (آ) وبسایت رسمی پروژهٔ آپاچی هدوپ. (ب) لینک پیشنهادی برای نصب هدوپ.

```
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADDOOP_HOME/lib/native
export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib/native"
```

سپس، جهت اعمال تغییرات بر روی محیط، از ویرایشگر خارج شده و دستور زیر را در ترمینال اجرا کنید:

```
$ source ~/.bashrc
```

در نهایت، لازم است نسخهٔ جاوای مورد استفاده را برای دستورات هدوپ تنظیم کرد. بدین منظور، به کمک یک ویرایشگر متن، فایل $HADOOP_HOME/etc/hadoop/hadoop-env.sh را باز کرده، خطی که به صورت export JAVA_HOME= می‌باشد را از حالت کامنت خارج کرده و آدرس فایل جی‌دی‌کی را مشابه شکل ۲.۲ در مقابل آن قرار دهید.



شکل ۲.۲: شمایی از فایل اجرایی محیط هدوپ.

برای اعمال تنظیمات مشابه در ویندوز، پاورشل را باز کرده و دستورات زیر را در آن وارد کنید:

```
$ SETX JAVA_HOME "path/to/jdk"  # required if you have not set
    java environment after installing java JDK
$ SETX HADOOP_HOME "path/to/hadoop-folder"
$ SETX PATH "$env:PATH;$env:JAVA_HOME/bin;$env:HADOOP_HOME/bin"
```

جهت اعمال تغییرات، پاورشل را بسته و مجدد باز کنید. جهت اطمینان از اعمال شدن تغییرات در هر یک از سیستم‌ها، دستور زیر را اجرا کنید:

```
$ hadoop version  # if the command failed in Windows PowerShell,
    try hadoop -version
Hadoop 3.3.1
Source code repository https://github.com/apache/hadoop.git -r
    a3b9c37a397ad4188041dd80621bdeefc46885f2
Compiled by ubuntu on 2021-06-15T05:13Z
Compiled with protoc 3.7.1
From source with checksum 88a4ddb2299aca054416d6b7f81ca55
This command was run using /home/bigdata/hadoop-3.3.1/share/
    hadoop/common/hadoop-common-3.3.1.jar
```

در صورتی‌که با خطا مواجه شدید و دستور شناخته نشد، مراحل بالا را مجدد و با دقت بررسی و تکرار کنید.

۲.۳.۲ نصب در حالت شبه‌توزیع‌شده

پس از طی مراحل بالا، هدوپ در حالت مستقل نصب شده و قابل استفاده است. جهت نصب در حالت شبه‌توزیع‌شده، لازم است تنظیماتی صورت گیرد که در ادامه به آن‌ها می‌پردازیم. این تنظیمات مستقل از سیستم عامل می‌باشند و تنها لازم است تغییرات لازم بر روی فایل‌های ذکر شده به کمک یک ویرایشگر متن اعمال شوند. توجه داشته باشید که آدرس‌های استفاده شده در این فایل‌ها با فرض وجود کاربری به نام `bigdata` در سیستم لینوکس است. آدرس‌ها را با توجه به سیستم خود و تنظیماتی که در مراحل قبل اعمال کرده‌اید، به‌طور مناسب تغییر دهید.

اولین گام در راستای نصب هدوپ در حالت شبه‌توزیع‌شده، تنظیم آدرس گره نام و آدرس فولدر موقتی که هدوپ در موتور نگاشت‌کاهش به‌کار خواهد گرفت، است. پس از ساختن فولدری بدین منظور (در اینجا فولدر `tmpdata` در خانه ساخته شده است)، فایل `core-site.xml` را باز کرده و تنظیمات زیر را در آن قرار دهید:

Listing 2.1: ”core-site.xml”

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
<property>
  <name>hadoop.tmp.dir</name>
  <value>/home/bigdata/tmpdata</value>
</property>
<property>
  <name>fs.default.name</name>
  <value>hdfs://127.0.0.1:9000</value>
</property>
</configuration>
```

همان‌طور که می‌بینید، دو خصوصیت در اینجا تعریف شده است. خصوصیت اول جهت تنظیم آدرس ذخیرهٔ موقت و خصوصیت دوم جهت تنظیم آدرس گره نام در خوشه است که در اینجا، درگاه ۹۰۰۰ میزبان محلی قرار داده شده است.

حال، لازم است تنظیمات مربوط به اچ‌دی‌اف‌اس، موتور نگاشت‌کاهش و یارن صورت پذیرند. در فایل مربوط به تنظیمات اچ‌دی‌اف‌اس، لازم است محل ذخیرهٔ گره‌های نام و داده و نیز، تعداد تکرارها را مشخص کنیم. تعداد تکرارها به‌صورت پیش‌فرض عدد ۳ خواهد بود. با این حال، از آنجایی که در حال نصب هدوپ در حالت شبه‌توزیع‌شده و بر روی یک گره هستیم، این عدد را برابر ۱ قرار خواهیم داد تا در مصرف حافظه صرف جویی شود. در تنظیمات موتور نگاشت‌کاهش نیز، کافیست مشخص کنیم که مدیریت منابع توسط یارن صورت خواهد گرفت و سپس، تنظیمات یارن را به‌طور مقتضی ایجاد کنیم. محتویات هر سه فایل در زیر آمده است:

Listing 2.2: ”hdfs-site.xml”

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
<property>
  <name>dfs.data.dir</name>
  <value>/home/bigdata/dfsdata/namenode</value>
</property>
<property>
```

```
  <name>dfs.data.dir</name>
  <value>/home/bigdata/dfsdata/datanode</value>
</property>
<property>
  <name>dfs.replication</name>
  <value>1</value>
</property>
</configuration>
```

Listing 2.3: ”mapred-site.xml”

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<configuration>
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
</configuration>
```

Listing 2.4: ”yarn-site.xml”

```
<?xml version="1.0"?>

<configuration>

<!-- Site specific YARN configuration properties -->
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</
   name>
  <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
```

```
  <name>yarn.resourcemanager.hostname</name>
  <value>127.0.0.1</value>
</property>
<property>
  <name>yarn.acl.enable</name>
  <value>0</value>
</property>
<property>
  <name>yarn.nodemanager.env-whitelist</name>
  <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,
    HADOOP_CONF_DIR,CLASSPATH_PERPEND_DISTCACHE,HADOOP_YARN_HOME,
    HADOOP_MAPRED_HOME</value>
</property>

</configuration>
```

در فایل تنظیمات یارن، مشاهده می‌کنید که یک سرویس جهت انجام فرآیند درهم‌آمیختن و مرتب‌سازی در موتور نگاشت‌کاهش تعریف شده است. همچنین، نام میزبان مدیر منابع و متغیرهای محیطی مورد استفاده توسط مدیر گره‌ها مشخص شده‌اند.

در این مرحله، سیستم شما در حالت شبه‌توزیع‌شده آماده است و می‌توانید به بخش ۴.۲ جهت راه‌اندازی هدوپ رجوع کنید.

۳.۳.۲ نصب در حالت تماماً‌توزیع‌شده

برای نصب هدوپ در یک خوشه با بیش از یک گره، لازم است مراحل توضیح داده شده در بخش ۱.۳.۲ را بر روی تمامی گره‌ها طی کنید. سپس، با در دست داشتن آی‌پی مربوط به هر گره و انتخاب یکی به‌عنوان گره ارباب، تنظیمات مربوط به فایل‌هایی که در حالت شبه‌توزیع‌شده دیدیم، به‌همراه فایل اجرایی محیط یارن، به صورت زیر خواهند بود (توجه کنید که آدرس فایل‌ها را به‌درستی و متناسب با هر گره اصلاح کنید.):

Listing 2.5: ”yarn-env.sh”

```
export  JAVA_HOME=/path/to/jdk
export  HADOOP_PREFIX=/path/to/hadoop-folder
export  PATH=$PATH:$HADOOP_PREFIX/bin:$JAVA_HOME/bin:.
export  HADOOP_COMMON_HOME=$HADOOP_PREFIX
```

```
export  HADOOP_HDFS_HOME=$HADOOP_PREFIX
export  YARN_HOME=$HADOOP_PREFIX
export  HADOOP_CONF_DIR=$HADOOP_PREFIX/etc/hadoop
export  YARN_CONF_DIR=$HADOOP_PREFIX/etc/hadoop
```

Listing 2.6: ”core-site.xml”

```
<property>
<name>fs.default.name</name>
<!-- replace Master-Hostname with the IP address of master node
   -->
<value>hdfs://Master-Hostname:9000</value>
</property>
<property>
<name>hadoop.tmp.dir</name>
<!-- make sure you have created the tmp directory -->
<value>/home/bigdata/hadoop-3.3.1/tmp</value>
</property>
```

Listing 2.7: ”hdfs-site.xml”

```
<property>
<name>dfs.replication</name>
<value>2</value>
</property>
<property>
<name>dfs.permissions</name>
<value>false</value>
</property>
```

Listing 2.8: ”yarn-site.xml”

```
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</
   name>
```

```
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
<name>yarn.resourcemanager.resource-tracker.address</name>
<value>Master-Hostname:8025</value>
</property>
<property>
<name>yarn.resourcemanager.scheduler.address</name>
<value>Master-Hostname:8030</value>
</property>
<property>
<name>yarn.resourcemanager.address</name>
<value>Master-Hostname:8040</value>
</property>
```

تنظیمات مربوط به موتور نگاشت‌کاهش مشابه آنچه در حالت شبه‌توزیع‌شده دیدیم، می‌باشد و در آن تنها یارن را به‌عنوان چارچوب مدیریت منابع معرفی می‌کنیم. در نهایت، فایل /$HADOOP_HOME etc/hadoop/slaves را ویرایش کرده و آی‌پی مربوط به گره‌های برده را در آن قرار می‌دهیم.

۴.۲ راه‌اندازی هدوپ

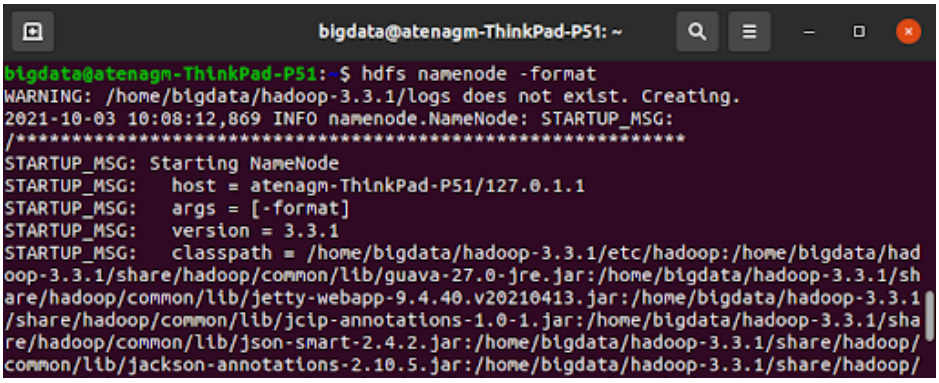
وقتی هدوپ را برای اولین بار راه‌اندازی می‌کنیم، لازم است گره نام را فرمت کنیم. بدین منظور، کافیست دستور زیر را در ترمینال و یا پاورشل اجرا کنید:

```
$ hdfs namenode -format
```

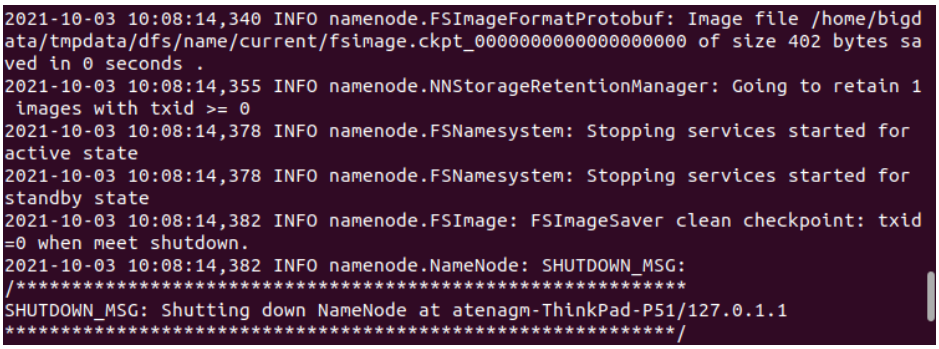
نمونه‌ای از نتیجهٔ اجرای دستورات بالا در شکل ۳.۲ قابل مشاهده است.

جهت آغاز سرویس‌های هدوپ می‌توانید از دستور start-all.sh استفاده کنید. این دستور تمامی سرویس‌های مرتبط را آغاز می‌کند. به‌منظور جلوگیری از آغاز فرآیندهای اضافی، می‌توانید تنها اچ‌دی‌اف‌اس و یارن را به‌کمک دستورات زیر آغاز کنید:

```
$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [atenagm-ThinkPad-P51]
$ start-yarn.sh
```



(آ)



(ب)

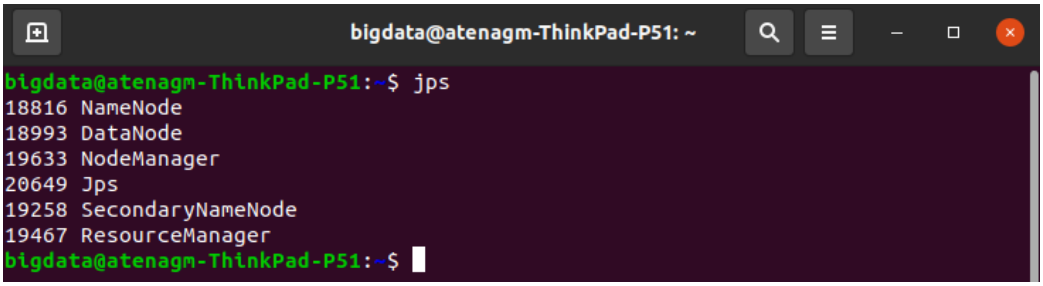
شکل ۳.۲: چند خط آغازین و پایانی حاصل از فرمت گره نام.

```
Starting resourcemanager
Starting nodemanagers
```

توجه کنید که حتماً پیغام آغاز گره نام، گره داده، گره نام ثانوی، مدیر منابع و مدیر گره مشاهده شوند. به‌منظور کسب اطمینان از فعال بودن تمامی دیمن‌ها و اجرای آن‌ها به‌عنوان فرآیندهای جاوا، از دستور `jps` استفاده کنید. این دستور، فرآیندهای در حال اجرای جاوا را لیست می‌کند (شکل ۴.۲). در صورتی‌که هر یک از گره‌های داده یا نام را در خروجی مشاهده نکردید، دستورات زیر را اجرا کنید:

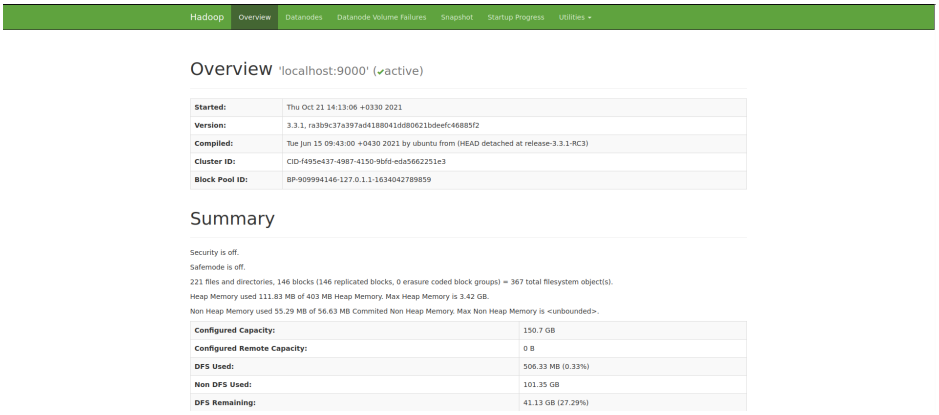
```
$ rm -r /tmp/hadoop-<username>
$ hdfs namenode -format
```

در این مرحله، می‌توانید به رابط گرافیکی هدوپ از طریق مرورگر وب دسترسی پیدا کنید. در

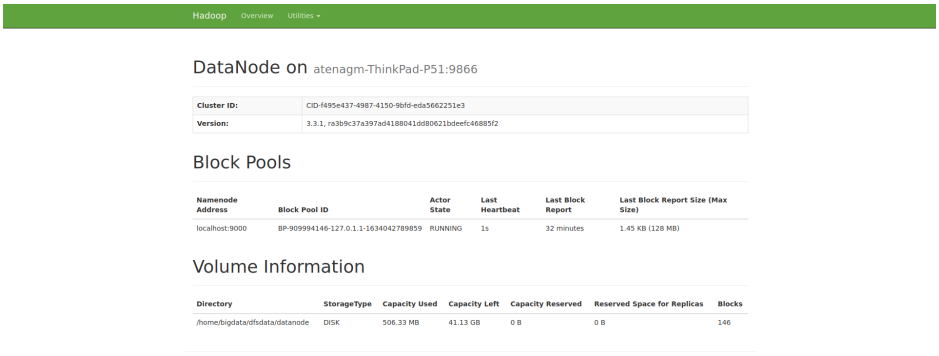


شکل ۴.۲: فرآیندهای در حال اجرای جاوا پس از آغاز اچ‌دی‌اف‌اس و یارن.

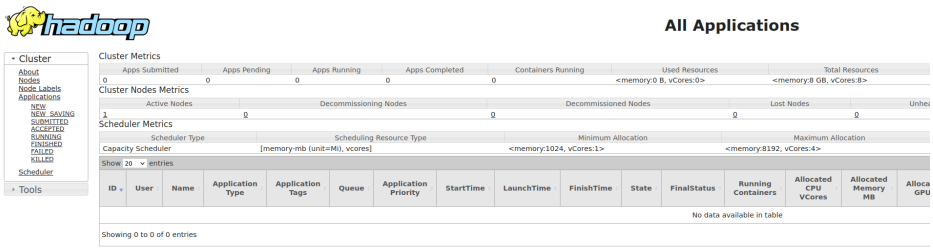
صورتی‌که هدوپ را در حالت تماماً‌توزیع‌شده نصب کرده باشید، این دسترسی از طریق آی‌پی گره‌ها امکان‌پذیر است. در حالت شبه‌توزیع‌شده، این دسترسی از طریق میزبان محلی خواهد بود. برای دسترسی به رابط گرافیکی گره نام، گره‌های داده و یارن می‌توانید به‌ترتیب از درگاه‌های ۹۸۷۰، ۹۸۶۴ و ۸۰۸۸ استفاده کنید. نمونه‌ای از اجرای رابط‌های گرافیکی را می‌توانید در شکل ۵.۲ مشاهده کنید. در انتها، برای بستن فرآیندهای در حال اجرا، می‌توانید در تمام دستوراتی که برای آغاز فرآیندها به‌کار بردید، به‌جای واژهٔ start از stop استفاده کنید.



(آ)



(ب)



(ج)

شکل ۵.۲: رابط گرافیکی گره نام (آ)، گره داده (ب) و یارن (ج).

فصل ۳

نحوه کار با اچ‌دی‌اف‌اس

برای تعامل با اچ‌دی‌اف‌اس، می‌توان از خط فرمان استفاده کرد. دستورات اچ‌دی‌اف‌اس مشابه دستورات یونیکس هستند و لذا کار با آن دشوار نیست. با این حال، در شرایطی ممکن است در حال نگارش برنامه‌ای سطح بالا مثل پایتون باشیم که در آن نیاز به تعامل با اچ‌دی‌اف‌اس و اجرای دستورات آن داریم. در این صورت، می‌توان از کتابخانه‌ای که برای این منظور فراهم شده است، استفاده کرد. در ادامه، پس از بررسی نحوهٔ تعامل با اچ‌دی‌اف‌اس از طریق خط فرمان، به معرفی این کتابخانه و نحوهٔ کار با آن می‌پردازیم. این فصل با توجه به تجربهٔ عملی و منابع [۱۱] و [۹] و مستندات کتابخانهٔ معرفی شده تنظیم شده‌اند.

۱.۳ نحوه تعامل از طریق محیط خط فرمان

تعامل با سیستم فایل‌های توزیع‌شده از طریق خط فرمان با یکی از دو کلیدواژه زیر امکان‌پذیر است:

```
$ hadoop fs -<command> [-option <arg>]
$ hdfs dfs -<command> [-option <arg>]
```

در ادامه، جهت سادگی، از دستور دوم استفاده شده است ولی توجه کنید که هر دو دستور به‌طور مشابه قابل استفاده‌اند و تفاوتی ندارند. همان‌طور که پیش‌تر اشاره شد، دستوراتی که جهت تعامل با سیستم فایل‌ها در هدوپ به‌کار می‌روند غالباً دستورات یونیکس هستند. به‌طور مثال، برای لیست کردن فایل‌های ریشه (دایرکتوری روت)، می‌توان از دستور `ls` به‌صورت زیر استفاده کرد:

```
$ hdfs dfs -ls /
Found 2 items
```

```
drwx------   - bigdata supergroup          0 2021-10-12 16:17 /
   tmp
drwxr-xr-x   - bigdata supergroup          0 2021-10-12 16:31 /
   user
```

همان‌طور که می‌بینید، در اینجا دو مورد یافت شده است. هر خط خروجی، به‌ترتیب شامل مجوزهای فایل/فولدر، تعداد تکرارهای فایل، کاربر فایل/فولدر، گروه آن، حجم آن به بایت، تاریخ و زمان ایجاد، و در نهایت نام فایل/فولدر است. دقت کنید که در مثال بالا، از آنجایی‌که هر دو مورد یافت شده فولدر هستند، تعداد تکرار با خط تیره نمایش داده شده است؛ زیرا تکرار تنها بر روی فایل‌ها اعمال می‌شود. همچنین، حجم محاسبه نشده و برابر با صفر گزارش شده است.

حال، فرض کنید مجموعه‌ای از دادگان را در اختیار داریم. می‌خواهیم فولدری در اچ‌دی‌اف‌اس ساخته و داده را در آن قرار دهیم. بدین منظور، به روش زیر عمل می‌کنیم:

```
# use mkdir to create an empty directory
$ hdfs dfs -mkdir webacc
# list the home directory to make sure the folder is created
$ hdfs dfs -ls
Found 1 items
drwxr-xr-x   - bigdata supergroup          0 2021-10-13 11:26
   webacc
# move a file named access_log to the folder we created
$ hdfs dfs -put access_log webacc/access_log
$ hdfs dfs -ls webacc
Found 1 items
-rw-r--r--   1 bigdata supergroup  504941532 2021-10-13 11:27
   webacc/access_log
```

اکنون، فایل مورد نظر بر روی اچ‌دی‌اف‌اس قرار گرفته است. در حالت کلی، جهت انتقال داده از حافظه داخلی به اچ‌دی‌اف‌اس و برعکس، می‌توان از دستورات زیر استفاده کرد:

```
# from local to hdfs:
$ hdfs dfs -put <local-source> <hdfs-destination>
# or alternatively:
$ hdfs dfs -copyFromLocal <local-source> <hdfs-destination>
# from hdfs to local:
$ hdfs dfs -get <hdfs-source> <local-destination>
# or alternatively:
```

```
$ hdfs dfs -copyToLocal <hdfs-source> <local-destination>
```

همان‌طور که مشاهده می‌کنید، تعداد تکرارهای فایل برابر با ۱ و حجم فایل در حدود ۵۰۵ مگابایت است. شاید کنجکاو باشید که بدانید آیا واقعاً فایل به صورت بلوک دخیره شده است یا خیر. به کمک ابزار بازرسی سیستم فایل‌ها، مشابه زیر، می‌توانید اطلاعاتی در خصوص فایل/فولدر مورد نظر کسب کنید:

```
$ hdfs fsck /user/bigdata/webacc/access_log -blocks
Connecting to namenode via http://localhost:9870/fsck?ugi=
   bigdata&blocks=1&path=%2Fuser%2Fbigdata%2Fwebacc%2Faccess_log
FSCK started by bigdata (auth:SIMPLE) from /127.0.0.1 for path /
   user/bigdata/webacc/access_log at Thu Oct 21 17:15:40 IRST
   2021


Status: HEALTHY
Number of data-nodes: 1
Number of racks:    1
Total dirs:     0
Total symlinks:   0

Replicated Blocks:
Total size: 504941532 B
Total files:  1
Total blocks (validated): 4 (avg. block size 126235383 B)
Minimally replicated blocks:  4 (100.0 %)
Over-replicated blocks: 0 (0.0 %)
Under-replicated blocks:  0 (0.0 %)
Mis-replicated blocks:    0 (0.0 %)
Default replication factor: 1
Average block replication:  1.0
Missing blocks:   0
Corrupt blocks:   0
Missing replicas:   0 (0.0 %)
Blocks queued for replication:  0

Erasure Coded Block Groups:
```

```
Total size: 0 B
Total files:  0
Total block groups (validated): 0
Minimally erasure-coded block groups: 0
Over-erasure-coded block groups:  0
Under-erasure-coded block groups: 0
Unsatisfactory placement block groups:  0
Average block group size: 0.0
Missing block groups:   0
Corrupt block groups:   0
Missing internal blocks:  0
Blocks queued for replication:  0
FSCK ended at Thu Oct 21 17:15:40 IRST 2021 in 1 milliseconds


The filesystem under path '/user/bigdata/webacc/access_log' is
   HEALTHY
```

در بخش مربوط به گزارش بلوک‌های تکرار شده، تعداد بلوک‌هایی که فایل در آن ذخیره شده است، ۴ گزارش شده است. همچنین، در مقابل آن، متوسط حجم بلوک‌ها بر حسب بایت نمایش داده شده است.

دستورات بالا، یک نسخهٔ کپی شده از فایل/فولدر را بر روی حافظهٔ داخلی و یا اچ‌دی‌اف‌اس ایجاد می‌کرد. حال، فرض کنید می‌خواهیم فایل/فولدر کاملاً منتقل شود و از روی محل مبدأ حذف شود. در این صورت، می‌توان از دستورات زیر بهره برد:

```
# move from local to hdfs
$ hdfs dfs -moveFromLocal <local-source> <hdfs-destination>
# move from hdfs to local
$ hdfs dfs -moveToLocal <hdfs-source> <local-destination>
```

همچنین، اگر بخواهیم فایل/فولدری را از روی اچ‌دی‌اف‌اس حذف کنیم، مشابه دستورات یونیکس، می‌توان به‌صورت زیر عمل کرد:

```
# remove single file
$ hdfs dfs -rm /path/to/file
# remove the whole directory
$ hdfs dfs -rm -r /path/to/folder
```

در اینجا، تعدادی از اساسی‌ترین دستورات را بررسی کردیم. بدیهی است که دستورات به موارد بالا محدود نمی‌شوند و همچنین، تسلط بر تمامی دستورات در گرو استفادهٔ چندین باره از آن‌هاست که ممکن است بسیاری از آن‌ها به‌صورت متداول استفاده نشوند. لذا، هر زمان که نیاز به یافتن و یادآوری دستوری بود، می‌توانید به یکی از دو صورت زیر، لیستی از دستورات مجاز، نحوهٔ استفاده از آن‌ها و نیز کاربرد آن‌ها تهیه کنید:

```
# get a detailed instruction
$ hdfs dfs -help
...
# seek a brief list of usage
$ hdfs dfs -usage
Usage: hadoop fs [generic options]
[-appendToFile <localsrc> ... <dst>]
[-cat [-ignoreCrc] <src> ...]
[-checksum [-v] <src> ...]
[-chgrp [-R] GROUP PATH...]
[-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
[-chown [-R] [OWNER][:[GROUP]] PATH...]
[-concat <target path> <src path> <src path> ...]
[-copyFromLocal [-f] [-p] [-l] [-d] [-t <thread count>] <
   localsrc> ... <dst>]
[-copyToLocal [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst
   >]
[-count [-q] [-h] [-v] [-t [<storage type>]] [-u] [-x] [-e] [-s]
    <path> ...]
[-cp [-f] [-p | -p[topax]] [-d] <src> ... <dst>]
[-createSnapshot <snapshotDir> [<snapshotName>]]
[-deleteSnapshot <snapshotDir> <snapshotName>]
[-df [-h] [<path> ...]]
[-du [-s] [-h] [-v] [-x] <path> ...]
[-expunge [-immediate] [-fs <path>]]
[-find <path> ... <expression> ...]
[-get [-f] [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
[-getfacl [-R] <path>]
[-getfattr [-R] {-n name | -d} [-e en] <path>]
[-getmerge [-nl] [-skip-empty-file] <src> <localdst>]
```

```
[-head <file>]
[-help [cmd ...]]
[-ls [-C] [-d] [-h] [-q] [-R] [-t] [-S] [-r] [-u] [-e] [<path>
   ...]]
[-mkdir [-p] <path> ...]
[-moveFromLocal [-f] [-p] [-l] [-d] <localsrc> ... <dst>]
[-moveToLocal <src> <localdst>]
[-mv <src> ... <dst>]
[-put [-f] [-p] [-l] [-d] [-t <thread count>] <localsrc> ... <
   dst>]
[-renameSnapshot <snapshotDir> <oldName> <newName>]
[-rm [-f] [-r|-R] [-skipTrash] [-safely] <src> ...]
[-rmdir [--ignore-fail-on-non-empty] <dir> ...]
[-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>]|[--set <
   acl_spec> <path>]]
[-setfattr {-n name [-v value] | -x name} <path>]
[-setrep [-R] [-w] <rep> <path> ...]
[-stat [format] <path> ...]
[-tail [-f] [-s <sleep interval>] <file>]
[-test -[defswrz] <path>]
[-text [-ignoreCrc] <src> ...]
[-touch [-a] [-m] [-t TIMESTAMP (yyyyMMdd:HHmmss) ] [-c] <path>
   ...]
[-touchz <path> ...]
[-truncate [-w] <length> <path> ...]
[-usage [cmd ...]]

Generic options supported are:
-conf <configuration file>        specify an application
   configuration file
-D <property=value>               define a value for a given
   property
-fs <file:///|hdfs://namenode:port> specify default filesystem
   URL to use, overrides 'fs.defaultFS' property from
   configurations.
-jt <local|resourcemanager:port>  specify a ResourceManager
-files <file1,...>                specify a comma-separated list
```

```
    of files to be copied to the map reduce cluster
-libjars <jar1,...>                 specify a comma-separated list
    of jar files to be included in the classpath
-archives <archive1,...>            specify a comma-separated list
    of archives to be unarchived on the compute machines

The general command line syntax is:
command [genericOptions] [commandOptions]
```

۲.۳ نحوه تعامل از طریق برنامه‌نویسی در زبان پایتون

شرکت اسپاتیفای[1] یک پکیج پایتون جهت ساختن یک کارخواه هدوپ و کار با اچ‌دی‌اف‌اس و دسترسی به آن داخل برنامه‌های پایتون ارائه داده است. این پکیج که اسنِیک‌بایت[2] نام دارد، به‌طور مشخص برای نسخه‌های دوم پایتون توسعه یافت و سپس، با گسترش روزافزون نسخهٔ سوم پایتون، یک نسخه ویژهٔ پایتون ۳ نیز ایجاد شد. اسنیک‌بایت شامل یک کتابخانه برای ایجاد کارخواه است که می‌تواند به‌طور مستقیم با گره نام ارتباط برقرار کند. برای نصب اسنیک‌بایت، از دستور زیر استفاده کنید:

```
$ pip install snakebite-py3
```

حال می‌توانید با استفاده از کتابخانهٔ `Client` و ساختن یک نمونه از آن، با گره نام ارتباط برقرار کنید. برای نمونه، به برنامهٔ زیر توجه کنید:

Listing 3.1: ”listdir.py”

```
from snakebite.client import Client


client = Client('localhost', 9000)
for x in client.ls(['/']):
  print(x)
```

همان‌طور که می‌بینید، برای ساخت یک کارخواه، آدرس گره نام در خوشه را به‌عنوان پارامترهای `Client` به آن داده‌ایم. پارامتر اول، آدرس آی‌پی میزبان و پارامتر دوم، درگاهی است که گره نام

[1] Spotify
[2] Snakebite

روی آن قرار دارد. از آنجایی که هدوپ را در حالت شبه‌توزیع‌شده نصب کرده‌ایم، تمامی گره‌ها و کارها بر روی میزبان محلی قرار دارند و همان‌طور که در تنظیمات هنگام نصب دیدیم، درگاهی که برای دسترسی به گره نام تعریف شد، درگاه ۹۰۰۰ است. کلاس `Client` پارامترهای متعددی نیز دارد که مهم‌ترین آن‌ها به شرح زیر می‌باشند:

- `host`: این پارامتر اجباری بوده و یک مقدار رشته‌ای، بیانگر نام میزبان یا آی‌پی آدرس گره نام، می‌گیرد.
- `port`: این پارامتر یک عدد صحیح و نمایانگر درگاه گره نام است. مقدار پیش‌فرض این پارامتر برابر با ۸۰۲۰ است.
- `hadoop_version`: این پارامتر یک عدد صحیح با مقدار پیش‌فرض ۹ است که نسخهٔ پروتکل هدوپ را نمایش می‌دهد (پروتکل هدوپ جهت مسلسل‌سازی داده به‌کار می‌رود).
- `use_trash`: این پارامتر بولی تعیین می‌کند که آیا هنگام انجام عملیات حذف داده از اچ‌دی‌اف‌اس از یک سطل زباله استفاده شود یا نه. در صورتی‌که از سطل زباله استفاده نشود، دادهٔ حذف شده قابل برگشت نخواهد بود. مقدار این پارامتر به‌طور پیش‌فرض `False` است.
- `effective_user`: این پارامتر، کاربر مؤثر هدوپ را تعیین می‌کند که به‌صورت پیش‌فرض، به کاربر فعلی اشاره می‌کند.

کلاس `Client` متدهای متنوعی دارد که هر کدام معادل با یکی از دستورات اچ‌دی‌اف‌اس که پیش‌تر دیدیم، عمل می‌کند. این متدها غالباً هم‌نام با دستورات متناظرشان در اچ‌دی‌اف‌اس هستند. برای مثال، در قطعه کد بالا از متد `ls` جهت تهیهٔ لیستی از محتویات مسیر(های) داده شده (در اینجا، مسیر ریشه) بهره گرفته شده است. در حقیقت، برنامهٔ listdir.py دیکشنری‌هایی حاوی اطلاعات فایل‌ها موجود در مسیر ریشه را چاپ می‌کند و خروجی نمونه به‌صورت زیر است:

```
{'file_type': 'd', 'permission': 448, 'path': '/tmp', 'length':
    0, 'owner': 'bigdata', 'group': 'supergroup', '
    block_replication': 0, 'modification_time': 1634042832921, '
    access_time': 0, 'blocksize': 0}
{'file_type': 'd', 'permission': 493, 'path': '/user', 'length':
     0, 'owner': 'bigdata', 'group': 'supergroup', '
    block_replication': 0, 'modification_time': 1634043696607, '
    access_time': 0, 'blocksize': 0}
```

برخی از مهم‌ترین متدهای این کلاس، به شرح زیر هستند:

- `ls(paths,recurse=False,include_toplevel=False,include_children=True)`

  این دستور لیستی از محتویات مسیرهای مشخص شده در `paths` تهیه می‌کند. سه پارامتر دیگر این، به‌ترتیب، تهیهٔ لیست به‌صورت بازگشتی، شمول مسیر داده شده در لیست تهیه شده و شمول فرزندان در تهیهٔ لیست را تعیین می‌کنند.

- `mkdir(paths, create_parent=False, mode=493)`

  این متد جهت ایجاد دایرکتوری در مسیرهای داده شده به‌کار می‌رود. `create_parent` تعیین می‌کند که آیا دایرکتوری‌های والد را در صورت عدم وجود، بسازد یا نه. پارامتر `mode` نیز حالت ساخت را تعیین می‌کند.

- `delete(paths, recurse=False)`

  این متد معادل دستور `rm` در اچ‌دی‌اف‌اس است و مسیرهای مشخص شده در `path` را حذف می‌کند. پارامتر `recurese` نیز جهت حذف بازگشتی تمام محتویات به‌کار می‌رود.

- `copyToLocal(paths, dst, check_crc=False)`

  این متد جهت انتقال فایل از اچ‌دی‌اف‌اس به حافظهٔ محلی به‌کار می‌رود و تمامی فایل‌های مشخص شده در `paths` را در مسیر `dst` کپی می‌کند. پارامتر آخر جهت بررسی اورهای احتمالی به‌کار می‌رود.

- `df()`

  اطلاعات سیستم فایل را به‌صورت یک دیکشنری چاپ می‌کند.

- `rename(paths, dst)`

  این متد مشابه دستور `mv` در یونیکس و اچ‌دی‌اف‌اس عمل می‌کند.

دو برنامهٔ زیر مثال‌هایی از حذف فایل و ساخت دایرکتوری می‌باشند:

Listing 3.2: ”makedir.py”

```
from snakebite.client import Client
```

```
client = Client('localhost', 9000)
for p in client.mkdir(['test/input', 'test/output'],
    create_parent=True):
  print(p)
```

Listing 3.3: ”deletedir.py”

```
from snakebite.client import Client


client = Client('localhost', 9000)
for p in client.delete(['test'], recurse=True):
  print(p)
```

در برنامهٔ makedir.py، از آنجایی‌که پارامتر create_parent ست شده است، اگر فولدر test که والد فولدرهای مورد نظر جهت ایجاد است، از پیش وجود نداشته باشد، ابتدا آن را ایجاد کرده و سپس به ساخت فولدرهای input و output در داخل آن می‌پردازد. در صورتی‌که create_parent=False و دایرکتوری test وجود نداشته باشد، خروجی مشابه زیر خواهد بود:

```
{'path': '/user/bigdata/test/input', 'result': False, 'error': "
    java.io.FileNotFoundException\nParent directory doesn't exist
    : /user/bigdata/test\n\tat org.apache.hadoop.hdfs.server.
    namenode.FSDirectory.verifyParentDir(FSDirectory.java:2009)
...
 org.apache.hadoop.security.UserGroupInformation.doAs(
    UserGroupInformation.java:1878)\n\tat org.apache.hadoop.ipc.
    Server$Handler.run(Server.java:2966)\n"}
{'path': '/user/bigdata/test/output', 'result': False, 'error':
    "java.io.FileNotFoundException\nParent directory doesn't
    exist: /user/bigdata/test\n\tat org.apache.hadoop.hdfs.server
    .namenode.FSDirectory.verifyParentDir(FSDirectory.java:2009)
...
 org.apache.hadoop.security.UserGroupInformation.doAs(
    UserGroupInformation.java:1878)\n\tat org.apache.hadoop.ipc.
    Server$Handler.run(Server.java:2966)\n"}
```

در حالیکه اگر برنامه را چنان‌که نوشته شده است، اجرا کنیم، خروجی به‌صورت زیر خواهد بود:

```
{'path': '/user/bigdata/test/input', 'result': True}
```

```
{'path': '/user/bigdata/test/output', 'result': True}
```

عبارت 'result': True حاکی از موفقیت در ایجاد دایرکتوری‌ها و عدم بروز خطا می‌باشد.

حال، اگر بخواهیم دایرکتوری test که پیش‌تر ایجاد کردیم حذف کنیم، کافیست برنامهٔ deletedir.py را اجرا کنیم. نتیجهٔ اجرای برنامه به‌صورت زیر خواهد بود:

```
{'path': '/user/bigdata/test', 'result': True}
```

اگر در این برنامه، پارامتر recurse را برابر با False قرار می‌دادیم، برنامه با خطایی مشابه آنچه در زیر آمده است، مواجه می‌شد:

```
Traceback (most recent call last):
File "deletedir.py", line 5, in <module>
for p in client.delete(['test'], recurse=False):
File "/home/bigdata/.local/lib/python3.8/site-packages/snakebite
   /client.py", line 514, in delete
for item in self._find_items(paths, processor, include_toplevel=
   True):
File "/home/bigdata/.local/lib/python3.8/site-packages/snakebite
   /client.py", line 1237, in _find_items
entry = processor(full_path, fileinfo.fs)
File "/home/bigdata/.local/lib/python3.8/site-packages/snakebite
   /client.py", line 513, in <lambda>
processor = lambda path, node, recurse=recurse: self.
   _handle_delete(path, node, recurse)
File "/home/bigdata/.local/lib/python3.8/site-packages/snakebite
   /client.py", line 520, in _handle_delete
raise DirectoryException("rm: `%s': Is a directory" % path)
snakebite.errors.DirectoryException: rm: `/user/bigdata/test':
   Is a directory
```

فصل ۴

نگاشت‌کاهش در پایتون

هدوپ به‌گونه‌ای طراحی شده است که به‌راحتی بتوان توابع نگاشت و کاهش را به صورت برنامه‌هایی به زبان جاوا نوشت. علاوه‌براین، هدوپ دارای ابزاری به‌نام جریان‌سازی هدوپ[1] است که امکان ساخت کارهای نگاشت‌کاهش در هر زبان دیگری را فراهم می‌کند. در این بخش، نحوهٔ نگارش توابع نگاشت و کاهش در پایتون را بر روی مثال‌های عملی بررسی می‌کنیم؛ ولی پیش از آن، لازم است مروری بر نحوهٔ عملکرد این توابع داشته باشیم.

همان‌طور که در فصل اول اشاره شد، تابع نگاشت، قطعه‌های دادهٔ ورودی را گرفته و پس از انجام پردازش‌های لازم، زوج‌هایی از کلید–مقدار را در خروجی تولید می‌کند. سپس، زوج‌های کلی–مقدار درهم آمیخته و مرتب شده و به تابع کاهش جهت انجام پردازش‌های بعدی داده می‌شوند. در حقیقت، این توابع ورودی‌های خود را از ورودی استاندارد[2] خط‌به‌خط دریافت کرده و خروجی‌ها را نیز خط‌به‌خط در خروجی استاندارد[3] می‌نویسند. هر خط خروجی تابع نگاشت، شامل یک زوج کلید–مقدار است که با تَب[4] از هم جدا شده‌اند. تابع کاهش نیز ورودی‌هایی با همین فرمت را دریافت می‌کند و خروجی‌هایی به همین شکل را در خروجی استاندارد می‌نویسد. ابزار جریان‌سازی هدوپ یک کار نگاشت‌کاهش بر اساس توابع نوشته شده تولید کرده، آن را بر روی خوشه قرار می‌دهد و تا پایان فرآیند، پیشرفت آن را دنبال می‌کند.

برای نگارش موتور نگاشت‌کاهش در زبان پایتون، دو راه کلی وجود دارد:

[1] Hadoop streaming
[2] stdin
[3] stdout
[4] Tab

۱. نوشتن یک فایل قابل اجرای پایتون،

۲. بهره‌گیری از کتابخانه‌هایی که امکان نوشتن توابع نگاشت و کاهش به‌صورت توابع مولد را فراهم می‌کنند.

در ادامه به بررسی هر دو روش خواهیم پرداخت. مطالب این بخش متاسب با تجربهٔ عملی، مستندات کتابخانه‌های معرفی شده و منابع [۹] و [۱۱] تهیه شده‌اند.

۱.۴ چگونگی نوشتن توابع نگاشت و کاهش

از آنجایی که توابع نگاشت و کاهش جهت انجام پردازش بر روی دادگان نوشته می‌شوند، با ارائهٔ چند مثال حقیقی به تشریح این بخش خواهیم پرداخت. مثال‌ها برگرفته از کورس آموزشی آشنایی با هدوپ و نگاشت‌کاهش در سایت یوداسیتی[۵] است و دادگان مورد استفاده، داده‌های مربوط به گزارش سرور وبسایت یک شرکت[۶] به‌صورت ناشناس است. هر خط از فایل داده، شامل اطلاعات ارسال یک درخواست به سرور این وبسایت است و متشکل از اطلاعاتی از جمله، آدرس آی‌پی درخواست‌کننده، تاریخ و زمان دسترسی و نام صفحه‌ای روی وبسایت است که به آن رجوع شده است. فرمت هر خط از داده به صورت زیر است:

```
ip id username [d/m/y:h:m:s zone] "mtd path qs/req" stat len
```

که در آن، `ip` آدرس آی‌پی کارخواه، `id` مشخصهٔ کارخواه، `username` نام کاربری کارخواه، `mtd` روش دسترسی، `path` مسیر درخواست شده، `qs/req` رشتهٔ پرس‌وجو و پروتکل درخواست، `stat` کد وضعیت درخواست و `len` اندازهٔ شیء برگشتی به بایت است. اطلاعات داخل [] نیز بیانگر تاریخ، زمان و حوزهٔ مکانی دسترسی است.

می‌خواهیم با بهره‌گیری از این فایل دادگان، به سؤالات زیر پاسخ دهیم:

۱. هر مسیر روی این وبسایت چند درخواست داشته است؟

۲. هر آدرس آی‌پی یکتا چند درخواست روی وبسایت ثبت کرده است؟

جهت پاسخگویی به هر یک از این سؤالات، لازم است خواسته‌ها را شناسایی کرده و فرمت خروجی مطلوب را مشخص کنیم. سپس، با تعیین فرمت ورودی و خروجی هر یک از توابع نگاشت و کاهش، به تعریف پردازش‌های لازم در هر یک از آن‌ها می‌پردازیم.

[5] Intro to Hadoop and MapReduce-Udacity

[۶] لینک دانلود دادگان

ابتدا با سؤال نخست آغاز می‌کنیم. هدف، محاسبهٔ تعداد درخواست به‌ازای هر مسیر روی وبسایت است. بنابراین، خروجی تابع کاهش به صورت زوج‌های (مسیر، تعداد درخواست) خواهد بود. ورودی تابع نگاشت خط‌های فایل داده به فرمت ذکر شده در بالا است. پس تابع نگاشت باید هر خط را پردازش کرده، مسیر را از آن استخراج کند و زوج‌های به شکل (مسیر،۱) تولید کند؛ زیرا در هر خط، مسیر تنها ۱ بار آمده است. سپس، در تابع کاهش، تمام زوج‌های با کلید یکسان را با یکدیگر تجمیع کرده و مجموع تکرارهای هر مسیر را محاسبه می‌کنیم. بدین ترتیب، تعداد کل درخواست‌ها برای هر مسیر را در دست خواهیم داشت. بنابراین، برنامه‌های نگاشت و کاهش به‌صورت زیر خواهند بود:

Listing 4.1: ”hit_mapper.py”

```
#!/usr/bin/env python3

import sys


for line in sys.stdin:
  data = line.strip().split("GET ")
  if len(data) > 1:
    filename = data[1].split()[0]
    print('{0}\t{1}'.format(filename, 1))
```

Listing 4.2: ”hit_reducer.py”

```
#!/usr/bin/env python3

import sys


total = 0
prev = None

for line in sys.stdin:
  data = line.strip().split("\t")

  if len(data) != 2:
    continue
```

```
  k, v = data
  if prev and prev != k:
    print('{0}\t{1}'.format(prev, total))
    total = 0

  prev = k
  total += int(v)

if prev:
  print('{0}\t{1}'.format(prev, total))
```

همان‌طور که پیش‌تر گفته شد، توابع نگاشت و کاهش با ورودی و خروجی استاندارد کار می‌کنند که در پایتون به‌کمک کتابخانهٔ sys قابل دسترسی هستند. در فایل hit_mapper.py، بر روی خطوط ورودی حرکت کرده، هر خط را با رشتهٔ " "GET تقسیم می‌کنیم. اگر پس از این رشته، داده‌ای وجود داشته باشد، کافیست آن رشته را با فواصل تقسیم کنیم. اولین زیررشتهٔ حاصل دربردارندهٔ مسیر درخواستی است. پس آن را به همراه عدد ۱ که یک تب با نام مسیر فاصله دارد، در خروجی چاپ می‌کنیم.

می‌دانیم که خروجی‌های حاصل از اجرای تابع نگاشت بر روی قطعه‌های مختلف داده، مرتب می‌شوند. بنابراین، تمامی درخواست‌های مربوط به یک مسیر پشت سر هم قرار گرفته‌اند. پس در فایل hit_reducer.py، پس از تفکیک هر خط از ورودی با تب و به‌دست آوردن نام مسیر و عدد مقابل آن، اگر نام فایل با نام فایلی که در خط پیشین خوانده‌ایم متفاوت باشد، بدین معنی است که دیگر درخواستی برای آن مسیر قبلی وجود نداشته است. پس کافیست نام مسیر را به‌همراه مجموع به‌دست آمده در خروجی چاپ کنیم.

در سؤال دوم خواسته شده است تا تعداد درخواست‌های هر کارخواه یکتا را بیابیم. بنابراین، خروجی تابع کاهش شامل آدرس آی‌پی هر کارخواه و تعداد درخواست‌های ارسال شده توسط او خواهد بود. این بار، جهت پردازش ورودی، باید آدرس آی‌پی را از هر خط استخراج کنیم و مشابه سؤال قبل عمل کنیم. لذا، فایل نگاشت به‌صورت زیر خواهد بود:

Listing 4.3: ”hit_per_ip_mapper.py”

```
#!/usr/bin/env python3

import sys
```

```
for line in sys.stdin:
  data = line.strip().split()
  if len(data) > 1:
    ip = data[0]
    print('{0}\t{1}'.format(ip, 1))
```

از آنجایی که آدرس آی‌پی اولین زیررشته در هر خط می‌باشد، به‌راحتی می‌توان آن را استخراج و زوج‌های (آدرس آی‌پی،۱) را تشکیل داد. نکتهٔ جالب آن است که نیازی به تابع کاهش جدید نیست و همان تابع قبلی قابل استفاده است.

هم‌اکنون، کدها از لحاظ تئوری درست هستند و تنها لازم است آن‌ها را بر روی دادگان داده شده تست کنیم و نتایج را ارائه گزارش کنیم. همواره توصیه می‌شود که پیش از تست موتور نگاشت‌کاهش بر روی خوشه و کل دادگان، بر روی کارخواه محلی و با بخش بسیار کوچکی از دادگان تست و بررسی شوند تا کدها دیباگ شوند و پس از اطمینان حاصل کردن از صحت کدها، منابع خوشه را در اختیار آن‌ها قرار دهیم. به همین منظور، جهت بررسی صحت کدهای مربوط به سؤال نخست، ابتدا دستور زیر را در خط فرمان اجرا می‌کنیم تا یک بازنمایش از اجرای موتور نگاشت‌کاهش داشته باشیم:

```
$ head -n 50 access_log | ./hit_mapper.py | sort | ./hit_reducer
    .py
```

دستور بالا، ابتدا ۵۰ خط اول فایل دادگان را در خروجی استاندارد قرار می‌دهد. سپس، فایل نگاشت را بر روی این نتایج اجرا کرده و پس از مرتب کردن خروجی آن، فایل کاهش را اجرا می‌کند. در این مرحله، می‌توانید با بررسی خروجی و خطاهای احتمالی، کد را همچون یک برنامهٔ عادی پایتون دیباگ کنید. اگر با خطای Permission denied مواجه شدید، احتمالاً کدهای شما مجوز اجرا ندارند. لذا، پس از اعطای مجوز اجرا به آن‌ها به‌کمک دستور زیر، دستور بالا را مجدداً اجرا کنید.

```
$ chmod a+x <filename>
```

حال که کد بدون نقص است، می‌توانیم آن را بر روی خوشه اجرا کنیم. پس از خارج کردن فایل دادگان از حالت فشرده و قرار دادن آن بر روی اچ‌دی‌اف‌اس به‌کمک دستوراتی که در فصل قبل آموختیم، جهت اجرای کار نگاشت‌کاهش از دستور زیر بهره می‌بریم:

```
$ hadoop jar HADOOP_HOME/share/hadoop/tools/lib/hadoop-streaming
    -*.har -files hit_mapper.py,hit_reducer.py -input webacc/
```

```
access_log -output webacc/hit_counts -mapper hit_mapper.py -
reducer hit_reducer.py
```

توجه کنید که در مستندات دستور hadoop jar آمده است که جهت اجرای یارن از دستور jar yarnlr استفاده کنید. با این حال، از آنجایی که در تنظیمات هنگام نصب، مدیریت منابع موتور نگاشت‌کاهش را به سمت اجرای یارن هدایت کردیم، دو دستور معادل عمل خواهند کرد. هر دوی دستورات به‌منظور اجرای ابزار جریان‌سازی هدوپ به‌کار می‌روند که در مسیر مشخص شده قرار دارند. پس از تعیین مسیر فایل جار مورد نظر، باید مسیر فایل‌های مربوط به کدهای نگاشت و کاهش را مشخص کنیم. سپس، با تعریف مسیری که داده در آن قرار دارد، مشخص کردن مسیری جهت ذخیرهٔ خروجی و تعیین فایل‌های نگاشت و کاهش، دستور را کامل می‌کنیم. همچنین، می‌توان با استفاده از گزینهٔ combiner-، یک تلفیق‌گر نیز مشخص کرد که عمدتاً همان فایل کاهش خواهد بود. به روش مشابه، می‌توان کدهای مربوط به سؤال دوم را نیز پس از اطمینان از صحت آن‌ها، اجرا کرد.

اگر تمامی مراحل به‌صورت گفته شده طی شده باشد، با خطایی مواجه نخواهید شد. به عنوان یک قانون کلی، همیشه به‌خاطر داشته باشید که اگر کدی را بدون تست بر روی محیط محلی بر روی اچ‌دی‌اف‌اس تست کردید و با خطا مواجه شدید، اولین قدم بازگشت به کد و دیباگ آن است. همچنین، همیشه مطمئن شوید که کدها اجازهٔ اجرا دارند. دستهٔ دیگری از مشکلات می‌توانند ناشی از تنظیمات زمان نصب، از جمله آدرس گره‌ها بر روی خوشه باشند. در متن خطاهایی از این جنس، صراحتاً ذکر می‌شود که آدرس اجراکنندهٔ کد با آدرس تنظیم شده مطابقت ندارد و غالباً از جنس ConnectException می‌باشند. این مشکل با بررسی مجدد تنظیمات رفع خواهد شد. اگر تناقضی در تنظیمات وجود نداشت، مجدداً گره نام را فرمت کنید (در فصل‌های پیشین به آن پرداخته شده است).

۲.۴ بهره‌گیری از کتابخانه‌ها

کتابخانه‌هایی جهت نگارش موتور نگاشت‌کاهش در پایتون و بدون نیاز به اجرای مجزای آن توسط جار، توسعه یافته‌اند که از جملهٔ آن‌ها می‌توان به هدوپای[۷]، پای‌دوپ[۸] و دیسکو[۹] و ام‌آرجاب[۱۰] اشاره کرد. با این حال، پروژه‌ای که به‌صورت فعال به‌روزرسانی می‌شود و کار با آن ساده می‌باشد،

[7] Hadoopy
[8] Pydoop
[9] Disco
[10] mrjob

ام‌آرجاب است که توسط شرکت یلپ[11] توسعه یافته است. به همین دلیل، این کتابخانه از محبوبیت بالایی برخوردار است و در اینجا نیز استفاده از آن را بررسی خواهیم کرد.

این کتابخانه به برنامه‌نویس این امکان را می‌دهد تا کارهای نگاشت‌کاهش خود را بر روی سیستم محلی تست کند و یا بر روی خوشه آن را اجرا کند. با این حال، از لحاظ عملکردی، در مقایسه با حالت پیشین، سرعت پایین‌تری دارد و در صورتی‌که زمان اجرا از اهمیت بالایی برخوردار باشد، استفاده از آن توصیه نمی‌شود. در ام‌آرجاب، نیازی به نوشتن فایل‌های جداگانه برای کارهای نگاشت و کاهش نیست؛ بلکه کافیست یک کلاس شامل متدهایی مولد برای انجام کارهای نگاشت و کاهش تعریف کنیم. کد زیر پاسخی برای سؤال نخست به‌کمک این کتابخانه است:

Listing 4.4: ”hit_count_mrjob.py

```
from mrjob.job import MRJob


class HitCount(MRJob):
  def mapper(self, _, line):
    data = line.strip().split("GET ")
    if len(data) > 1:
      filename = data[1].split()[0]
      yield(filename, 1)

  def reducer(self, filename, counts):
    yield(filename, sum(counts))


if __name__ == '__main__':
  HitCount.run()
```

به‌منظور نوشتن موتور نگاشت‌کاهش به‌کمک این کتابخانه، کافیست کلاسی بنویسیم که از کلاس `MRJob` ارث‌بری می‌کند. در اینجا خبری از حلقه نیست، چراکه توابع به‌صورت مولدهایی به‌ازای هر خط نگارش می‌شوند و متد `run` خود، صدا زدن توابع را مدیریت می‌کند. توابع نگاشت، کاهش و تلفیق‌گر با متدهایی با نام‌های `mapper`، `reducer` و `combiner` هستند که هر کدام از آن‌ها دو پارامتر ورودی دارند. این دو پارامتر همان زیررشته‌هایی هستند که با تب جدا شده‌اند. با این حال، ورودی تابع نگاشت غالباً یک خط بدون تب است که در پارامتر دوم قرار می‌گیرد.

[11] Yelp

برنامهٔ نوشته شده به‌کمک این کتابخانه می‌تواند در حالت‌های گوناگونی اجرا شود:

- `-r inline`

  این گزینه جهت اجرای کد به صورت یک فرآیند پایتون بر روی سیستم محلی به‌کار می‌رود.

- `-r local`

  با استفاده از این گزینه، می‌توان برنامه را به‌صورت چند زیرفرآیند پایتون بر روی سیستم محلی اجرا کرد و تا حدی اجرا بر روی اچ‌دی‌اف‌اس را شبیه‌سازی کرد.

- `-r hadoop`

  این گزینه برای اجرای برنامه روی هدوپ و اچ‌دی‌اف‌اس استفاده می‌شود.

بنابراین، جهت اجرای برنامه‌ای که در بالا نوشتیم بر بستر هدوپ، می‌توان از دستور زیر کمک گرفت:

```
$ python hit_count_mrjob.py -r hadoop hdfs:///user/bigdata/
   webacc/access_log
```

فصل ۵

جمع‌بندی و نتیجه‌گیری

در این گزارش به بررسی مفاهیم اولیهٔ هدوپ و اجزاء آن، از جمله اچ‌دی‌اف‌اس، موتور نگاشت‌کاهش و یارن پرداختیم. همچنین، تاریخچه‌ای از آغاز و توسعهٔ هدوپ به‌همراه توضیح مختصری از پروژه‌های مرتبط را ارائه کردیم. دیدیم که هدوپ چارچوبی متن‌باز با دسترس‌پذیری بالا است که با قرارگیری در کنار دیگر نرم‌افزارهای اکوسیستم هدوپ، امکانات گسترده‌ای را جهت پردازش و تحلیل مه‌داده‌ها در اختیار قرار می‌دهد.

سپس، به نصب و کار با هدوپ پرداخته شد. پس از بررسی حالت‌های نصب، توضیح مبسوطی جهت نصب در حالت مستقل و شبه‌توزیع‌شده بر روی سیستم‌های مبتنی بر یونیکس، به‌ویژه لینوکس/دبیان، ارائه شد. راهنمای نصب بر روی سیستم عامل ویندوز و نیز نصب در حالت تماماً توزیع‌شده تشریح شده است. با ارائهٔ راهنما جهت راه‌اندازی اچ‌دی‌اف‌اس و یارن، به بررسی دستورات جهت تعامل با اچ‌دی‌اف‌اس از طریق خط فرمان و یا داخل برنامه‌های پایتون پرداختیم. در نهایت، با بررسی مثالی عملی، نگارش توابع نگاشت و کاهش در زبان پایتون را مورد مطالعه قرار دادیم.

هدف این کتابچه، آشنایی اولیه با مفاهیم و مبانی کار با هدوپ بود و همچنان مباحث زیادی از دنیای گستردهٔ هدوپ ناگفته مانده است. در ادامه، منابعی را برای افراد علاقه‌مند معرفی می‌کنیم تا با مطالعهٔ آن‌ها، بتوانند در این دنیای گسترده گام‌های مؤثرتری بردارند.

- یکی از منابع اصلی این گردایه، کتاب هدوپ از تام وایت بود [۱۱]. تام وایت یکی از توسعه‌دهندگان پروژهٔ هدوپ است و توضیحات دقیقی از نحوهٔ کار با اچ‌دی‌اف‌اس، نگاشت‌کاهش و یارن ارائه کرده است. همچنین، به بررسی برخی پروژه‌های مرتبط نیز پرداخته است. با وجود اینکه این کتاب برای نسخهٔ دوم هدوپ نگارش شده است، همچنان منبع جامعی برای افراد علاقه‌مند است.

- کتاب Hadoop in Practice نوشتهٔ الکس هولمز از نشر Manning از جمله کتاب‌هایی است که به افراد علاقه‌مند جهت یادگیری عملی تحلیل داده‌ها و مه‌داده‌ها به‌کمک هدوپ معرفی می‌شود. این کتاب با ارائهٔ مثال‌های متعدد یکی از کتاب‌های کمکی خوب به‌شمار می‌رود.
- برای آن دسته از افرادی که می‌خواهند بیشترین بهره را از آخرین نسخهٔ هدوپ ببرند، توصیه می‌شود نگاهی به کتاب Big Data Analytics with Hadoop 3.0 نوشتهٔ سریدار آلا از نشر Packt داشته باشند. این کتاب با تکیه بر آخرین تغییرات صورت گرفته در پروژهٔ هدوپ، به تشریح تحلیل داده به‌کمک هدوپ و ابزارهایی نظیر اسپارک، پایتون و آر می‌پردازد.

کتاب‌نامه

[1] https://cwiki.apache.org/confluence/display/HADOOP2/Home.

[2] Apache hadoop official website. https://hadoop.apache.org/.

[3] What is hadoop? https://aws.amazon.com/emr/details/hadoop/what-is-hadoop/.

[4] https://www.geeksforgeeks.org/hadoop-version-3-0-whats-new/, 2020.

[5] Apache hadoop architecture explained (with diagrams). https://phoenixnap.com/kb/apache-hadoop-architecture-explained, 2020.

[6] R. Buyya, R. N. Calheiros, and A. V. Dastjerdi. *Big data: principles and paradigms*, chapter 7. Morgan Kaufmann, 2016.

[7] W. L. Bounsi. How to install hadoop on a real cluster (fully distributed mode). https://medium.com/@wilcoln/how-to-install-hadoop-on-a-real-cluster-fully-distributed-mode-7c3bc3fa7d7e, 2019.

[8] A. Mumtaz. A beginner's guide to hadoop's fundamentals. https://towardsdatascience.com/a-beginners-guide-to-hadoop-s-fundamentals-8e9b19744e30, 2021.

[9] Z. Radtka and D. Miner. *Hadoop with Python*. O'Reilly Media, 2015.

[10] S. Singhal. Hadoop : How to install in 5 steps in windows 10. https://medium.com/analytics-vidhya/hadoop-how-to-install-in-5-steps-in-windows-10-61b0e67342f8, 2021.

[11] T. White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 2015.