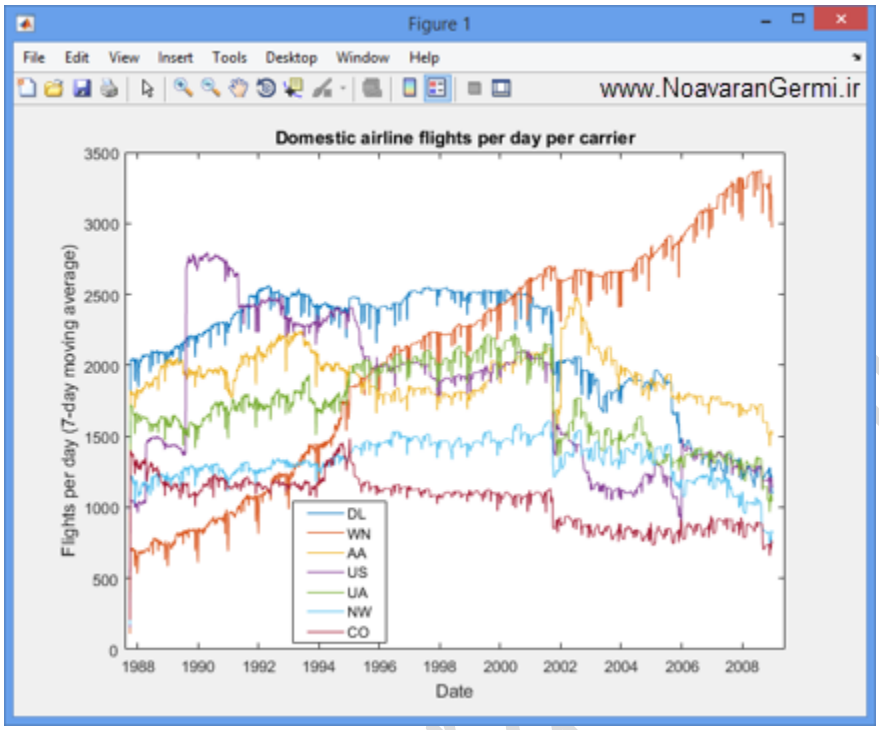
پردازش داده های خطوط هوایی (Big data) با نرم افزار MATLAB



این بخش به تکنیک برنامه‌نویسی MapReduce که در نسخه R2014b شبیه ساز MATLAB (متلب) در دسترس است، اختصاص دارد MapReduce .راهی را برای پردازش مقادیر بزرگ داده‌های مبتنی بر فایل بر روی کامپیوتر فراهم می‌کند. در شرایطی که مجموعه داده‌ها بسیار بزرگ باشد نیز کد MATLAB نوشته شده با استفاده از MapReduce می‌تواند بر روی پلتفرم "داده‌های بزرگ"، Hadoop، اجرا شود.

داده‌ها

مجموعه داده‌ای که استفاده خواهیم کرد شامل رکوردهایی حاوی معیارهای کارایی پروازهای خانگی خطوط هوایی آمریکا طی سال‌های ۱۹۸۷ تا ۲۰۰۸ است. هر سال دارای یک فایل جداگانه می‌باشد. اگر تجربه کار با داده‌های بزرگ را پیش از این داشته باشید، با این مجموعه داده آشنا هستید. مجموعه کامل داده از طریق این سایت قابل دانلود است .زیرمجموعه‌ی کوچکی از مجموعه داده، به نامairlinesmall.csv ، در MATLAB® در نظر گرفته شده تا شما بتوانید تمام مثال‌ها را بدون دانلود مجموعه داده کامل اجرا کنید.

آشنایی باmapreduce

Mapreduceیک تکنیک برنامه‌نویسی است که برای "تقسیم و غلبه" داده‌های بزرگ استفاده می‌شود. در متلب، تابع mapreduceنیازمند سه آرگومان ورودی است:

۱, یک datastore برای خواندن داده درون تابع "map" به صورت چانک به چانک.

۲, یک تابع "map" که بر روی هر چانک داده عمل می‌کند. خروجی این تابع یک محاسبه جزئی است . Mapreduceتابع map را یک بار برای هر چانک داده موجود در datastore فراخوانی می‌کند که هر عملیات به صورت مستقل از فراخوانی‌های دیگر map است.

۳, تابع "reduce" که به آن خروجی‌های حاصل از تابع map داده می‌شود. تابع reduce محاسبات شروع شده توسط تابع map را به پایان می‌رساند و پاسخ نهایی را تولید می‌کند.

توجه کنید که در اینجا قصد ما یک معرفی ساده بوده است. در واقع، خروجی یک فراخوانی تابع map پیش از تحویل داده شدن به تابع reduce می‌تواند هم خورده و به طرق جالبی ترکیب شود. این امر جلوتر بررسی خواهد شد.

استفاده از mapreduce برای انجام یک محاسبه

بیایید به مثالی برای به تصویر کشیدن چگونگی کارکرد mapreduce نگاهی بیندازیم. در این مثال می‌خواهیم طولانی‌ترین زمان پرواز در بین تمامی رکوردهای پرواز موجود در مجموعه داده کامل خطوط هوایی را پیدا کنیم. برای این کار باید موارد زیر را انجام دهیم:

۱, ایجاد یک شیء datastore برای مجموعه داده خطوط هوایی

۲, ایجاد یه تابع map که زمان پرواز ماکزیموم در هر چانک داده از datastore را محاسبه کند.

۳, ایجاد یک تابع reduce که ماکزیموم مقدار بین تمام ماکزیموم‌های محاسبه شده توسط تابع map را محاسبه کند.

ایجاد یک datastore :

datastoreبرای دسترسی به فایل‌های متنی جدولی (tabular) که روی یک دیسک محلی یا Hadoop® Distributed File System (HDFS™)نگهداری می‌شوند استفاده می‌شود. این مکانیزم همچنین در تهیه داده به صورت چانک به چانک برای فراخوانی‌های تابع map در زمان استفاده از mapreduceبه کار می‌رود.

بیایید یک datastore را ایجاد کرده و ابتدا مجموعه داده‌ها را مرور کنیم. این مرور باعث می‌شود که یک دید کلی از داده‌ها داشته باشیم و فرمت داده‌ها و ستون‌های حاوی داده‌هایی که به آنها علاقه‌مندیم را شناسایی کنیم . Previewاغلب یک چانک کوچک داده که حاوی تمام ستون‌های حاضر در مجموعه داده است را برای ما ایجاد می‌کند. در اینجا برای سادگی، تنها چند ستون را نمایش می‌دهیم.

```
ds = datastore('airlinesmall.csv', 'DatastoreType', 'tabulartext', ...
    'TreatAsMissing', 'NA');
data = preview(ds);
data(:,[1,2,3,9,12])
```

```
ans =
    Year    Month    DayofMonth    UniqueCarrier    ActualElapsedTime
    ____    _____    __________    _____________    _________________
    1987    10       21            'PS'              53
    1987    10       26            'PS'              63
    1987    10       23            'PS'              83
    1987    10       23            'PS'              59
    1987    10       22            'PS'              77
    1987    10       28            'PS'              61
    1987    10        8            'PS'              84
    1987    10       10            'PS'             155
```

برای محاسبه‌ای که می‌خواهیم انجام دهیم تنها نیاز است به ستون "ActualElapsedTime" نگاهی بیندازیم: این ستون حاوی اطلاعات دقیق زمان پرواز است. بیاییدdatastore مان را پیکربندی کنیم تا تنها این ستون را برای تابعmap مان فراهم کند.

```
ds.SelectedVariableNames = {'ActualElapsedTime'};
```

ایجاد یک تابعmap

اکنون تابع map را خواهیم نوشت .(MaxTimeMapper.m) سه آرگومان ورودی برای این تابع map باید فراهم شوند:

۱, داده ورودی، "ActualElapsedTime"، که به صورت یک جدول MATLAB توسط datastore تهیه می‌شود.

۲, مجموعه‌ای از اطلاعات پیکربندی و مفهومی، info. این آرگومان می‌تواند در بیشتر موارد از جمله اینجا نادیده گرفته شود.

۳, یک شیء برای ذخیره داده میانی، جایی که نتایج محاسبات تابع map در آن ذخیره شود. از تابع add برای اضافه کردن جفتِ کلید/مقدار به این خروجی میانی می‌توان استفاده کرد. در این مثال، ما نام این کلید را به صورت دلخواه 'MaxElapsedTime' گذاشته‌ایم.

تابع map ما مقدار ماکزیموم را در جدول 'data' پیدا کرده و یک کلید تک ('MaxElapsedTime') و مقدار مرتبط با آن را در شیء ذخیره داده میانی نگهداری می‌کند. اکنون جلوتر رفته و تابع map مقابل (MaxTimeMapper.m)را به پوشه (فولدر) کنونی‌مان اضافه می‌کنیم.

```
function MaxTimeMapper(data, ~, intermediateValuesOut)
maxTime = max(data{:,:});
add(intermediateValuesOut, 'MaxElapsedTime', maxTime);
end
```

ایجاد یک تابع reduce

قدم بعدی ایجاد تابع reduce است .(MaxTimeReducer.m) سه آرگومان ورودی نیز برای این تابع باید توسط mapreduce تهیه شود:

۱, یک مجموعه از کلیدهای ورودی. کلیدها در جلوتر بررسی می‌شوند، هرچند می‌توانند در بعضی مسائل ساده از جمله اینجا، در نظر گرفته نشوند.

۲, یک شیء برای ذخیره داده میانی که mapreduce آن را به تابع reduce ارسال می‌کند. این داده، خروجی تابع map است و در قالب جفت‌های کلید/مقدار می‌باشد. ما از توابع hasnext و getnext برای جستجو در میان مقادیر استفاده می‌کنیم.

۳, یک شیء برای ذخیره داده خروجی نهایی که نتایج محاسبات reduce در آن ذخیره می‌شوند. از توابع addو addmulti برای اضافه کردن جفت‌های کلید/مقدار به خروجی می‌توان استفاده کرد.

تابع reduce فهرستی از مقادیر مرتبط با کلید'MaxElapsedTime' که توسط فراخوانی‌های تابع map تولید شده‌اند، دریافت می‌کند. تابع reduce بین این مقادیر جستجو می‌کند تا ماکزیموم را بیابد .ما تابع reduceمعرفی شده در ادامه را ایجاد کرده (MaxTimeReducer.m) و آن را در پوشه کنونی ذخیره می‌کنیم.

```
function MaxTimeReducer(~, intermediateValuesIn, finalValuesOut)
maxElapsedTime = -inf;
while hasnext(intermediateValuesIn)
   maxElapsedTime = max(maxElapsedTime, getnext(intermediateValuesIn));
end
add(finalValuesOut, 'MaxElapsedTime', maxElapsedTime);
end
```

اجرایmapreduce

وقتی توابع map و reduce نوشته شده و در پوشه ذخیره شدند، می‌توانیم mapreduce را فراخوانی کرده و بهdatasore ، تابعmap ، و تابع reduce برای اجرای محاسباتمان بر روی داده‌ها ارجاع دهیم . تابع readallدر اینجا برای نمایش نتایج الگوریتم MapReduce استفاده شده است.

```
result = mapreduce(ds, @MaxTimeMapper, @MaxTimeReducer);
readall(result)
```

```
Parallel mapreduce execution on the local cluster:
********************************
*      MAPREDUCE PROGRESS      *
********************************
Map   0% Reduce   0%
Map 100% Reduce 100%
ans =
          Key           Value
    ________________    ______
    'MaxElapsedTime'    [1650]
```

اگر تولباکس Parallel Computing در دسترس باشد، متلب به صورت خودکار اجرای توابع map را موازی می‌کند. از آنجا که تعداد فراخوان‌های تابع map توسط mapreduce متناظر با تعداد چانک‌های datastoreاست، اجرای موازی فراخوان‌های map سرعت اجرا را به صورت کلی افزایش می‌دهد.

استفاده از کلیدها درmapreduce

استفاده از کلیدها یک ویژگی مهم mapreduce است. هر فراخوان تابع map می‌تواند نتایج میانی را به یک یا بیشتر "bucket"که کلید نامیده می‌شوند، بیفزاید.

اگر تابع map مقادیری را به چندین کلید اضافه کند، این امر منجر به چندین فراخوانی تابع reduce می‌شود. این کار باعث کاهش تعداد فراخوان‌هایی می‌گردد که در حال کار با مقادیر میانی تنها یک کلید هستند. تابع mapreduceبه صورت خودکار تمام این جابه‌جایی‌های داده بین فازهای map و reduce از الگوریتم را مدیریت می‌کند.

محاسبه معیارهای گروهیmapreduce

رفتار تابع map در این قضیه از تابع reduceپیچیده‌تر است و نشان‌دهنده مزایای استفاده از چند کلید برای ذخیره داده میانی می‌باشد. برای هر خط هوایی یافته شده در داده ورودی، از تابع addبرای افزودن برداری از مقادیر استفاده کنید. این بردار تعداد پروازهای روزانه طی ۲۱ سال را برای هر شرکت هوایی نشان می‌دهد. کد هر شرکت، کلید این بردار مقادیر است. این قضیه تضمین می‌کند که کلیه داده‌های هر شرکت هوایی با یکدیگر در یک گروه قرار گرفته و mapreduce گروه کامل را به تابع reduce تحویل می‌دهد.

تابع map جدید ما به این شکل است:(CountFlightsMapper.m)

```
function CountFlightsMapper(data, ~, intermediateValuesOut)
dayNumber = days((datetime(data.Year, data.Month, data.DayofMonth) - datetime(1987,10,1)))+1;
daysSinceEpoch = days(datetime(2008,12,31) - datetime(19875,10,1))+1;


[airlineName, ~, airlineIndex] = unique(data.UniqueCarrier);


for i = 1:numel(airlineName)
   dayTotals = accumarray(dayNumber(airlineIndex==i), 1, [daysSinceEpoch, 1]);
   add(intermediateValuesOut, airlineName{i}, dayTotals);
end
end
```

تابع reduce کمتر پیچیده است. این تابع به سادگی بین مقادیر میانی گشته و بردارها را به هم اضافه می‌کند. به محض تکمیل، مقادیر درون این بردار تجمعی را به عنوان خروجی بیرون می‌دهد. دقت کنید که تابع reduce نیازی به مرتب‌سازی یا امتحان مقادیر intemediateKeysInندارد؛ هر فراخوانی تابع reduce توسط mapreduceتنها مقادیر یک خط هوایی را منتقل می‌کند.

تابع جدید reduce ما بدین قرار است:(CountFlightsReducer.m)

```
function CountFlightsReducer(intermediateKeysIn, intermediateValuesIn, finalValuesOut)
daysSinceEpoch = days(datetime(2008,12,31) - datetime(1987,10,1))+1;
dayArray = zeros(daysSinceEpoch, 1);


while hasnext(intermediateValuesIn)
   dayArray = dayArray + getnext(intermediateValuesIn);
end
add(finalValuesOut, intermediateKeysIn, dayArray);
end
```

برای اجرای این تحلیل جدید، تنها کافی است datastore را ری-ست کرده و مقادیر موردعلاقه را انتخاب کنید. در اینجا ما تاریخ (سال، ماه،روز) و نام شرکت هوایی را می‌خواهیم. وقتی توابع map و reduceنوشته شده و در پوشه کنونی ذخیره شدند، mapreduceبهdatastore ، تابعmap ، و تابع reduce بروزرسانی شده ارجاع می‌دهد.

```
reset(ds);
ds.SelectedVariableNames = {'Year', 'Month', 'DayofMonth', 'UniqueCarrier'};

result = mapreduce(ds, @CountFlightsMapper, @CountFlightsReducer);
result = result.readall();
```

```
Parallel mapreduce execution on the local cluster:
********************************
*      MAPREDUCE PROGRESS      *
********************************
Map   0% Reduce   0%
Map 100% Reduce  50%
Map 100% Reduce 100%
```

تا اینجا تنها مجموعه داده نمونه را تحلیل کردیم .(airlinesmall.csv) برای دیدن تعداد پروازهای روزانه در کل مجموعه داده بیایید نتایج اجرای الگوریتم جدید MapReduce را روی کل مجموعه داده ببینیم.

```
load airlineResults
```

نمایش نتایج

پیش از نگاه به تعداد پروازهای روزانه هفت شرکت اول، بگذارید ابتدا فیلتری را روی داده انجام دهیم تا اثر سفرهای آخر هفته را کمی ملایم کنیم. در غیر این صورت، نمودار نتایج دچار آشفتگی زیادی خواهد شد.

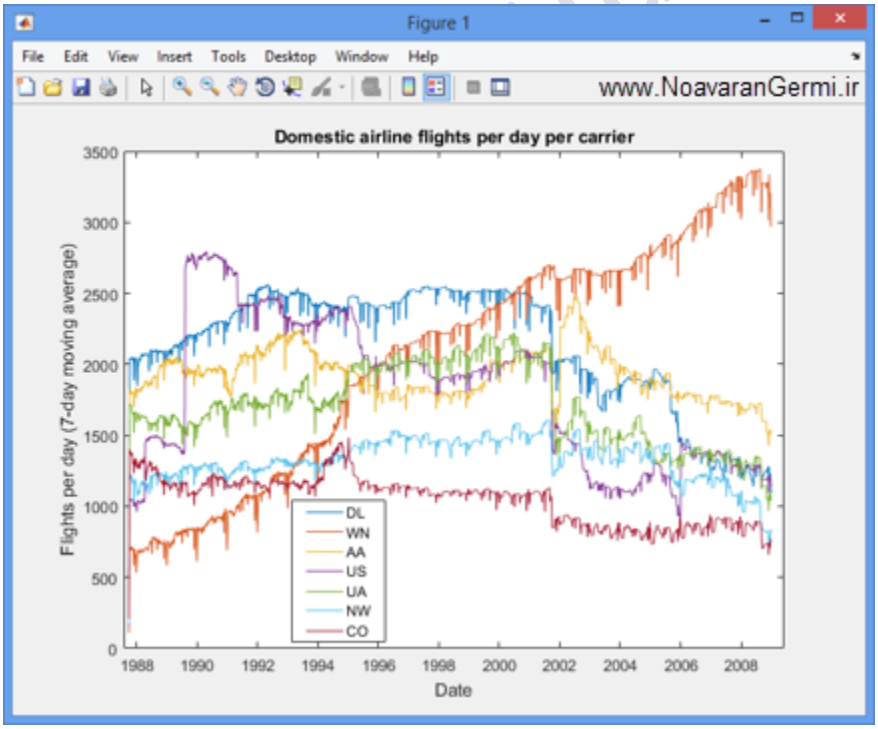
```
lines = result.Value;
lines = horzcat(lines{:});
[~,sortOrder] = sort(sum(lines), 'descend');
lines = lines(:,sortOrder(1:7));
result = result(sortOrder(1:7),:);

lines(lines==0) = nan;
for carrier=1:size(lines,2)
    lines(:,carrier) = filter(repmat(1/7, [7 1]), 1, lines(:,carrier));
end



figure('Position',[1 1 800 800]);
plot(datetime(1987,10,1):caldays(1):datetime(2008,12,31),lines)
title ('Domestic airline flights per day per carrier')
xlabel('Date')
ylabel('Flights per day (7-day moving average)')

try
    carrierList = readtable('carriers.csv.txt', 'ReadRowNames', true);
    result.Key = cellfun(@(x) carrierList(x, :).Description, result.Key);
catch
end
legend(result.Key, 'Location', 'SouthOutside')
```



.در طول این بازه زمانی است(WN) Southwest Airlines نکته جالب نمودار، رشد خط هوایی

اجرای mapreduce روی Hadoop

کدی که اکنون ایجاد کردیم بر روی کامپیوتر شخصی بود و به ما اجازه تحلیل داده‌ای که به صورت عادی درون حافظه ماشین ما جا نمی‌شد را به ما داد. اما چه می‌شود اگر داده ما بر روی پلتفرم داده‌های بزرگ، Hadoop، ذخیره شده بود و اندازه آن بسیار بزرگتر از آن بود که بتواند به کامپیوتر ما منتقل شود؟

با استفاده از تولباکس Parallel Computing (محاسبات موازی) بر روی MATLAB® Distributed Computing Server™، کد ایجاد شده توسط ما می‌تواند بر روی یک کلاستر راه دور Hadoop اجرا شود. توابع mapو reduce بدون تغییر باقی می‌مانند، اما دو تغییر در پیکربندی باید انجام شود:

۱, datastore ما برای ارجاع به مکان فایل‌های داده در Hadoop® Distributed File System (HDFS™)بروزرسانی می‌شود.

۲, شیء mapreducer برای ارجاع به Hadoop به عنوان محیط اجرا، بروزرسانی می‌شود.

بعد از اینکه این تغییرات داده شدند، الگوریتم می‌تواند روی Hadoop اجرا شود.