Strategies for Analyzing a 12-Gigabyte Data Set: Airline Flight Delays

Giới thiệu

Các công nghệ sử dụng

- Ngôn ngữ lập trình R
- Lệnh Shell và pipes (đường ống)
- Cơ sở dữ liệu quan hệ (SQL)
- Tính toán song song

Tập dữ liệu đầu vào

Dữ liệu đầu vào được lấy từ trang web http://stat-computing.org/dataexpo/2009/the-data.html. Trang web có 21 file CSV được nén bằng định dạng bunzip2 (bz2), chứa dữ liệu thu thập được trong 21 năm, từ năm **1987** tới năm **2008**, bao gồm 29 biến dữ liệu. Thông tin cụ thể của tập dữ liệu như sau:

STT	Biến	Mô tả
1	`Year`	Năm (1987-2008)
2	`Month`	Tháng (1-12)
3	`DayofMonth`	Ngày (1-31)
4	`DayOfWeek`	Ngày trong tuần: 1(thứ hai) - 7(chủ nhật)
5	`DepTime`	Thời gian khởi hành thực tế (theo giờ địa phương, hhmm)
6	`CRSDepTime`	Thời gian khởi hành dự kiến (theo giờ địa phương, hhmm)
7	`ArrTime`	Thời gian đến thực tế (theo giờ địa phương, hhmm)
8	`CRSArrTime`	Thời gian đến dự kiến (theo giờ địa phương, hhmm)
9	`UniqueCarrier`	
10	`FlightNum`	Số chuyến bay
11	`TailNum`	Số ở đuôi máy bay
12	`ActualElapsedTime`	Thời gian bay thực tế
13	`CRSElapsedTime`	Thời gian bay dự kiến
14	`AirTime`	
15	`ArrDelay`	Thời gian đến trễ (phút)
16	`DepDelay`	Thời gian khởi hành trễ (phút)
17	`Origin`	Mã sân bay khởi hành, mã gồm ba chữ cái, được định nghĩa bởi Hiệp hội Vận tải Hàng không Quốc tế (IATA)
18	`Dest`	Mã sân bay hạ cánh, mã gồm ba chữ cái, được định nghĩa bởi Hiệp hội Vận tải Hàng không Quốc tế (IATA)
19	`Distance`	Khoảng cách bay (dặm)
20	`TaxiIn`	
21	`TaxiOut`	
22	`Cancelled`	Có phải chuyến bay bị hủy: 0(Sai) - 1(Đúng)
23	`CancellationCode`	Nguyên do hủy chuyến: A(Carrier) - B(Lý do thời tiết) - C(NAS) - D(Lý do an ninh)
24	`Diverted`	Chuyển hướng bay: 0(Sai) - 1(Đúng)
25	`CarrierDelay`	(phút)
26	`WeatherDelay`	Trễ do thời tiết (phút)
27	`NASDelay`	Trễ do Hệ thống Hàng không Quốc gia (NAS) quy định (phút)

STT	Biến	Mô tả
28	`SecurityDelay`	Trễ do an ninh (phút)
29	`LateAircraftDelay`	Máy bay đến trễ (phút)

Giới thiệu các công nghệ sử dụng

- R là một ngôn ngữ lập trình mạnh mẽ để giải quyết các bài toán thống kê, trong bài toán này ta sử dụng cấu trúc dữ liệu bigmatrix thuộc thư viện bigmemory để giải quyết bài toán dữ liệu lớn. bigmatrix hoạt động giống matrix trong R, tuy nhiên nó sẽ quản lý RAM một cách hiệu quả khi giải quyết bài toán dữ liệu lớn bằng cách
- UNIX Shell là một công cụ
-

Tiền xử lý dữ liệu:

Dữ liệu tải về được lưu trong 21 file được nén bởi định dạng bunzip2, ta sử dụng UNIX Shell - một công cụ đầy mạnh mẽ với cú pháp đơn giản - để giải nén những file dữ liệu.

```
#Shell:
for year in {1987..2008}
do
  bunzip2 $year.csv.bz2
done
```

- Do cấu trúc dữ liệu bigmatrix yêu cầu tất cả giá trị phải cùng một kiểu dữ liệu, do đó ta sẽ ánh xạ những giá trị dạng chữ bằng một con số, cách đơn giản nhất là tận dụng kiểu dữ liệu factor trong R. Ý tưởng của chúng ta sẽ là sử dụng `read.csv()` để đọc và lưu dữ liệu vào một biến, sau đó ánh xạ và ghi đè dữ liệu đó trở lại file CSV.
- Tuy nhiên để tối ưu hóa bộ nhớ RAM, mỗi khi ta sử dụng xong biến đó thì ta phải giải phóng biến và gọi trình dọn dẹp bộ nhớ. Nguyên do là khi RAM bị sử dụng quá dung lượng hiện có, dữ liệu sẽ được chuyển sang ổ đĩa và được gọi lại RAM khi cần thiết. Và ta biết rằng tốc độ đọc từ ổ đĩa sẽ chậm hơn so với RAM, việc này sẽ khiến cho việc xử lý cực kỳ chậm chạp khi xử lý những dữ liệu lớn.
- Hàm để giải phóng biến là `rm()`, và hàm dọn dẹp bộ nhớ là `gc()` trong R

```
#R:
for(year in 1987:2008) {
  # stringAsFactors = TRUE sẽ chuyển tập dữ liệu dạng chuỗi sang kiểu factor
  x <- read.csv(paste(year, ".csv", sep = ""), stringAsFactors = TRUE)
  # Ta có năm biến dữ liệu trên được lưu ở dạng chuỗi
  x$UniqueCarrier <- unclass(x$UniqueCarrier)
  x$TailNum <- unclass(x$TailNum)
  x$Origin <- unclass(x$Origin)
  x$Dest <- unclass(x$Dest)
  x$CancellationCode <- unclass(x$CancellationCode)
  # Sau khi xử lý xong, ta ghi đè vào file CSV cũ
  write.csv(x, paste(year, ".csv", sep = ""), row.names = FALSE)
  # Ta giải phóng biến x
```

```
  rm(x)
  # Và gọi trình dọn dẹp bộ nhớ
  gc()
}
```