kdb+性能测试

笔者：陈泽天

日期：2019-07-23

横向速度测试

目的

测试 kdb+在行数相同的情况下，当列数增多的时候，csv 文件导入，hdb（历史数据库）导出，hdb 导入，平均值计算用时所产生的变化。

操作

每次生成一张 1，000，000 行的表格，列数分别为 10+10 列、10+20 列、10+50 列、10+100 列。前 10 列分别为日期（date）、小时（hour）、分钟（minute）、股票代码（ticker）、开盘价（op）、最高价（hp）、最低价（lp）、收盘价（cp）、股数（volume）、总额（amount）。这些数据都将尽可能根据实际情况随机生成。其余每列数据为从 10000～1000000 之间的随机浮点数。每次计算每行随机数的平均数，并记录各项用时。

具体过程

首先，编写 python 脚本去生成 csv 文件。脚本如下：

```
import random

if __name__ == "__main__":
    # Number of rows that are totally random
    rand_rows_num = 10
    # Number of records in the table
    records_num = 10

    with open('fakedata.csv', 'w') as f:
        # Write the column names
        f.write("date,hour,minute,ticker,op,hp,lp,cp,volume,amount")
        for row in range(0, rand_rows_num):
            f.write(",random{}".format(row))
        f.write("\n")

        # Write the data
        for _ in range(records_num):
            # Randomly if the transaction happens in the morning or afternoon
            morning = random.randint(0, 1)
            if morning == 0:  # Afternoon, time period = [13:01, 15:00]
                hour = random.randint(13, 15)
                if hour == 13:
                    minute = random.randint(1, 59)
                elif hour == 14:
                    minute = random.randint(0, 59)
                else:
                    minute = 0
            else:  # Morning, time period = [9:31, 11:30]
                hour = random.randint(9, 11)
                if hour == 9:
                    minute = random.randint(31, 59)
                elif hour == 10:
                    minute = random.randint(0, 59)
                else:
                    minute = random.randint(0, 30)

            # ticker is a 6-digit code
            # from Shenzheng (SZ), Shanghai (SH), Hong Kong (HK)
            ticker_number = random.randint(0, 999999)
            ticker_index = random.randint(0, 2)
            ticker = "{:06d}.{}".format(ticker_number, ["SZ", "SH", "HK"][ticker_index])
```

```
            # Let lowest price and highest price be within [10, 90]
            prices = [random.uniform(10, 90) for _ in range(2)]
            hp = max(prices)
            lp = min(prices)

            # Let opening price and closing price be within the lowest price and highest price
            op = random.uniform(lp, hp)
            cp = random.uniform(lp, hp)

            # Let volume be within [1000000, 10000000]
            volume = random.randint(1000000, 10000000)
            # Let amount be amount = volume * some random number between the lowest price and
highest price
            amount = volume * random.uniform(lp, hp)

            # Write the values into the csv file
            f.write("20190603,{},{},{},{},{},{},{},{}".format(hour, minute, ticker, op, hp,
lp, cp, volume, amount))
            # Randomly generate floats
            for _ in range(rand_rows_num):
                value = random.uniform(10000, 1000000)
                f.write(", {}".format(value))
            f.write("\n")
    f.close()
    print("Done.")
```

末尾