한국정보전자통신기술학회논문지(jkiiect)'20-8, Vol.13 No.4
논문 20-13-04-306 http://dx.doi.org/10.17661/jkiiect.2020.13.4.306

파이썬 활용한 데이터 처리 성능 향상방법 제안

김효관* 황원용**

Proposal For Improving Data Processing Performance Using Python

Hyo-Kwan Kim* Won-Yong Hwang**

요 약 본 논문은 대량의 데이터를 활용한 모델 개발 시 다양한 라이브러리를 갖춘 파이썬 언의의 성능 향상방법을 다룬다. 파이썬 언어는 엑셀과 같은 스프레드시트 형태 데이터 처리 시 Pandas 라이브러리를 사용한다. 데이터 처리 시 파이썬은 기가단위 이하 데이터 처리 시에는 인-메모리로 연산하여 성능 측면에서 크게 이슈가 없다. 하지만 기가단위 이상 데이터 처리 시 성능 이슈가 발생한다. 이에 본 논문은 데이터 처리 시 Pandas와 같이 사용할 수 있는 Dask 라이브러리를 활용하여 단일 클러스터 및 다중 클러스터에서 실행 작업을 분산처리 가능한 방법을 소개한다. 실험은 동일 사양의 하드웨어에서 간단한 지수산출 모델을 Pandas만 사용해서 처리하는 속도와 Dask를 같이 사용해서 처리하는 속도를 비교한다. 본 논문은 파이썬의 장점인 다양한 라이브러리를 쉽게 사용할 수 있다는 점을 유지하면서 성능측면에서도 대량의 데이터를 CPU 코어들이 분산 처리하여 모델을 개발할 수 있는 방법을 제시한다.

Abstract This paper deals with how to improve the performance of Python language with various libraries when developing a model using big data. The Python language uses the Pandas library for processing spreadsheet-format data such as Excel. In processing data, Python operates on an in-memory basis. There is no performance issue when processing small scale of data. However, performance issues occur when processing large scale of data. Therefore, this paper introduces a method for distributed processing of execution tasks in a single cluster and multiple clusters by using a Dask library that can be used with Pandas when processing data. The experiment compares the speed of processing a simple exponential model using only Pandas on the same specification hardware and the speed of processing using a dask together. This paper presents a method to develop a model by distributing a large scale of data by CPU cores in terms of performance while maintaining that python's advantage of using various libraries is easy.

Key Words : Python, Large scale of data, Pandas, Dask, Distributed processing. Cluster, Library

1. 서론

데이터 분석 과정은 다양한 데이터 소스를 수집한 후, 전처리 작업을 거쳐 통계적 모델이나 머신러닝 모델을 활용하여 분석한 후 의미 있는 결과를 만들어내는 과정이다. 파이썬 언어는 데이터분석 및 인공지능 분야에서 필요한 다양한 데이터 수집, 처리 및 분석 라이브러리(Selenium, Pandas, Seaborn, Matplotlib, Tensorflow, Keras 등)를 갖추고 있다. 라이브러리가 다양한 경우 개발자가 개발 아이디어만 있다면 Randomforest와 같은 머신러닝 알고리즘이나 웹 데이터를 수집하는 기능 등을 도서관에서 필요한 정보만 빌려서 쓰듯이 기능을 사용할 수 있다. 필요한 라이브

* Corresponding Author: Department of Fintech Korea Polytechnics (haiteam@kopo.ac.kr)
** Department of Fintech, Korea Polytechnics
Received July 16, 2020 Revised August 05, 2020 Accepted August 05, 2020

러리를 잘 검색해서 설치한 후 개발 시 가져다 쓰기만 하면 된다. 따라서 파이썬 언어는 데이터 분석 및 인공지능 영역에서 개발자가 가장 선호하는 언어이다. [그림 1]은 데이터 경진대회가 열리는 Kaggle 사이트에서 조사한 자료로 설문자의 83%가 파이썬(83%)을 분석 시 기본언어로 사용 하는 것으로 조사되었다. [1]

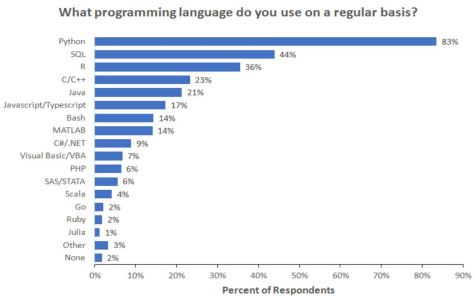

그림 1. Kaggle 선호언어 조사표
Fig. 1. Survey of Programming Language in Kaggle

하지만 파이썬은 메모리로 처리한다는 장점은 있으나 1기가 이상의 데이터를 처리하는데 있어서는 성능 이슈가 존재해 왔다. 특히 빅데이터 시대에 더 이상 샘플데이터 만으로 분석하지 않고 전체 데이터를 다루어야 하는 요구가 생겼으며, 신속하게 정보를 제공해야하는 필요성이 생겼다. [그림 2]와 같이 분석한 결과는 빠르게 필요한 곳에 전달되어야 데이터를 의미 있게 사용된다는 점을 시사한다.

그림 2. 데이터 분석 속도의 중요성
Fig. 2. Importance of Data Analysis Performance

따라서 본 논문에서는 파이썬에서 Apache Spark와 같은 분산처리가 가능한 Dask를 소개하고 Pandas만 사용해서 모델링하는 경우와 Dask를 같이 적용해서 모델링한 경우의 성능 비교를 통해 향후 파이썬의 데이터 처리 방법이 나아가야할 방향을 제시한다.[2]

2. 파이썬 전처리 기술

파이썬은 데이터 분석 시 가장 많이 사용되는 스프레드시트 형태의 데이터를 다루는 Pandas 라이브러리가 존재한다. 또한 대량의 데이터 처리 시 분산처리가 가능한 Dask를 같이 적용하면 개발한 파이썬 코드를 유지하면서 대량의 데이터도 파이썬으로 신속하게 처리할 수 있다.

2.1 Pandas

Pandas는 2008년 1월 출시된 스프레드시트 형태의 데이터를 조작하는데 사용되는 라이브러리다. 데이터 불러오기/저장하기 외 데이터 정제, 집계 및 병합 등 다양한 데이터 처리가 가능하다. [3]

그림 3. Pandas 소개
Fig. 3. Overview of pandas

2.2 Dask

Dask는 2018년 10월 출시된 파이썬에서 데이터 처리 작업을 병렬화하기 위한 잘 알려지지 않은 최신 프레임워크다. 멀티 코어가 있는 경우 연산 작업을 분산할 수 있다. Dask가 제공하는 기능이 분산처리 프레임워크인 Apache Spark와 유사하지만 Numpy, Pandas 및 Sklearn 등의 라이브러리와 통합되어 있다. [4] 따라서 개발자 입장에서는 쉽고 빠르게 기존 코드에 Dask를 적용하여 개발 모델 성능을 최적화 할 수 있다.

몇몇 개발자가 기존에 성능 이슈가 발생하면 분산처리가 가능한 Apache Spark를 적용하여 기존코드를

변환하여 처리하였다면 파이썬만 가지고도 분산처리 기술을 사용할 수 있다는 점이 새로운 점이다. [그림 4]는 Dask 사용법으로 Pandas나 Sklearn 의 사용법과 유사하다. Dask는 모든 복잡한 내부 요소를 추상화하고 파이썬 내에서 기존 Pandas로 구현된 코드를 병렬화 하는 API를 제공한다.

```
# Dataframes implement the Pandas API
import dask.dataframe as dd
df = dd.read_csv('s3://.../2018-*-*.csv')
df.groupby(df.account_id).balance.sum()
```

```
# Dask-ML implements the Scikit-Learn API
from dask_ml.linear_model \
  import LogisticRegression
lr = LogisticRegression()
lr.fit(train, test)
```

그림 4. Dask 사용법
Fig. 4. Use of Dask

3. 시스템 및 모델 정의

3.1 시스템 사양

실험에 사용할 분석 모델은 동일한 시스템 환경에서 성능 테스트를 수행한다. [그림 5]는 샘플 모델을 구동할 주요 PC 사양이다.

표 1. 시스템 환경구성
Table 1. System Environment

No	Div	Spec
1	OS	Windows 10 Pro 64-bit
2	CPU	Intel Core i7 @ 2.70Ghz (코어 2개)
3	RAM	16.0GB
4	Graphics	NVIDIA GeForce GTX 950M

[표 1]과 같이 코어 개수는 2개로 병렬처리 테스트가 가능한 환경에서 모델을 구동한다.

3.2 입력데이터

입력 데이터는 [표 2]와 같이 지역, 상품, 연주차별 과거 판매실적 정보를 대상으로 한다.

표 2. 데이터 설명
Table 2. Data Description

No	Column(ENG)	Column (KOR)	Desc
1	REGIONID	지역ID	A01
2	REGIONNAME	지역명	KOPO
3	AP1ID	지역서브ID	AP101
4	AP1NAME	지역서브명	EAST
5	ACCOUNTID	거래처ID	SALES0001
6	ACCOUNTNAME	거래처명	서울은행 1지점
7	SALESID	거래처서브ID	SALESID0001
8	SALESNAME	거래처서브명	서울은행 1지점
9	PRODUCTGROUP	제품군	PG01
10	PRODUCT	제품	PRODUCT01
11	ITEM	모델명	ITEM00001
12	YEARWEEK	연주차 정보	202001
13	YEAR	연도 정보	2020
14	WEEK	주차 정보	02
15	QTY	판매량	100
16	TARGET	판매 기대값	120

본 논문에서 사용하는 데이터는 크기만 다른(0.1GB / 1.5GB) 동일 칼럼의 판매실적 데이터를 사용한다. 데이터 크기 변화에 따라 Pandas만 활용 했을 때의 모델 처리속도와 Dask를 같이 적용 했을 때의 처리속도를 비교한다.

REGIONID	REGIONNAME	AP1ID	AP1NAME	ACCOUNTID	ACCOUNTNAME	SALESID	SALESNAME
A01	KOPO	401737	EAST	SALESID0017	서울은행17지점	SALESID0017	서울은행17 지점
A01	KOPO	401737	EAST	SALESID0017	서울은행17지점	SALESID0017	서울은행17 지점

PRODUCTGROUP	PRODUCT	ITEM	YEARWEEK	YEAR	WEEK	QTY	TARGET	IDX
PG4	PRODUCT12	ITEM02950	201728	2017	28	0	0.0	SALESID0017_ITEM02950
PG4	PRODUCT12	ITEM01161	201728	2017	28	0	0.0	SALESID0017_ITEM01161
PG4	PRODUCT12	ITEM02951	201728	2017	28	0	0.0	SALESID0017_ITEM02951
PG4	PRODUCT12	ITEM04036	201728	2017	28	0	0.0	SALESID0017_ITEM04036

그림 5. 데이터 예시
Fig. 5. Data Example

3.3 모델 프로세스

입력 데이터를 사용하여 구현할 샘플 모델은[그림 6]과 같다. 첫 번째 단계는 과거 제품의 판매실적 데이터를 불러온다. 성능 테스트를 위해 크기가 다른 데이터(0.1GB, 1.5GB)를 사용한다. 두 번째 단계는 과거 판매량 칼럼 값에 음수 값이 존재하는 경우 모두 0으로

정제한다. 실제 마이너스 값은 반품 수량으로 전처리 시 0 값으로 변경하는 단계이다. 이후 데이터를 기본키(지역, IDX, 연주차 칼럼)기준 오름차순 정렬한다. 세 번째 단계는 그룹(지역, IDX) 별 판매 추세선 값을 이동평균법(구간:5, 데이터 중심)을 사용하여 생성한다. 각 지역별 평균 판매추세 대비 실 판매량을 나누어 그룹, 연주차별 판매 지수를 산출한 후 결과를 저장한다.

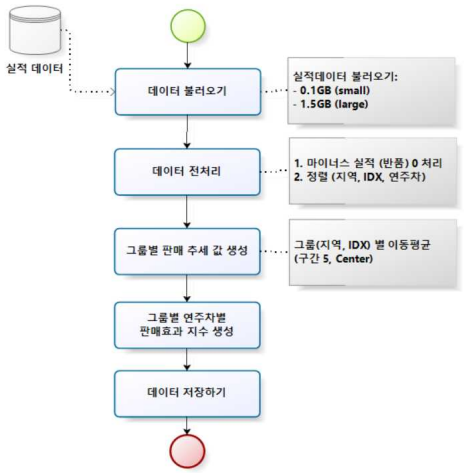

그림 6. 모델 설명
Fig. 6. Model Description

4. 모델 구현

분석 모델은 판매 실적 데이터를 사용하여 지역, IDX, 연주차별 판매 추세 값을 생성한 판매지수를 산출한다. 판매지수는 과거 연주차별로 동일 기간대의 평균 판매 추세 대비 각 연주차에 얼마나 잘 팔렸는지를 나타내는 연주차 효과 지수이다. 예를 들어 [그림 7]과 같이 미국 지역에 마우스가 2020년 3주차에 300개 팔렸는데 2020년 3주차 기준 5개 주차의 평균 판매량(100, 200, 300, 200, 100)이 180인 경우 2020년 3주차는 1.67배(180/300) 팔렸다는 지수를 생성하는 단계이다.

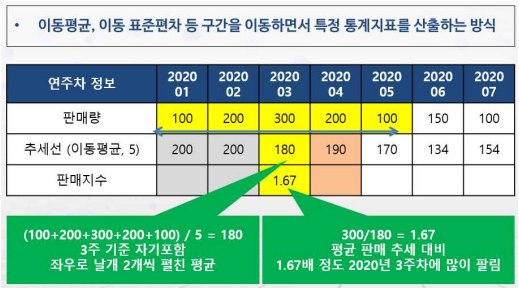

그림 7. 연주차효과지수 산출 단계
Fig. 7. Extract Yearweek-Effect Stage

4.1 데이터 전처리

주요 전처리 코드는 [그림 8]과 같다.

```
### 0. 데이터 불러오기
selloutDataS = pd.read_csv\
        ("../dataset/kopo_channel_result_small.csv",
           encoding="ms949")
print("data shape is {}".format(selloutDataS.shape))


### 1. 데이터 전처리: 반품수량 0 처리
def minusToZero(inValue):
    if inValue < 0:
        inValue = 0
    return inValue

selloutDataS["QTY"] = selloutDataS.QTY.apply(minusToZero)

### 2. 데이터 전처리: 정렬 ["REGIONID","IDX","YEARWEEK"]
sortKey = ["REGIONID","IDX","YEARWEEK"]
selloutDataS.sort_values(sortKey, inplace=True)
```

그림 8. 데이터 전처리 단계
Fig. 8. Data Preprocessing Stage

4.2 그룹별 판매 추세값 생성

그룹별 판매 특성만 반영한 추세 선을 생성하기 위하여 지역, IDX 별 그룹 키를 선언한 후 그룹별 판매량에 대해 이동평균 함수를 [그림 9]과 같이 적용한다. [5][6]

```
### 3. 그룹별 이동평균
groupKey = ["REGIONID","IDX"]

def maFunc(eachgroup):
    eachgroup["MA"] = eachgroup.QTY.\
                        rolling(window=5,
                                min_periods=1,
                                center=True).mean()
    return eachgroup[["REGIONID","IDX","YEARWEEK","QTY","MA"]]

finalResult = selloutDataS.groupby(groupKey).apply(maFunc)
```

그림 9. 추세선 생성 단계
Fig. 9. Extract Trend-line

4.3 Dask 변환

Pandas만 사용하여 구현한 코드에 Dask를 적용한다. 변환 시 기존 Pandas에 alias를 pd로 정의했다면 (import pandas as pd) dask는 dask.dataframe as dd 로 정의한다. 단 기본 라이브러리가 아니기 때문에 명령 창에서 pip install dask를 실행하여 라이브러리를 설치해야 한다. csv 데이터를 불러오는 경우 동일한 read_csv 함수를 사용 가능하다.[7]

```
import dask.dataframe as dd

selloutDataS = dd.read_csv\
    ("../dataset/kopo_channel_result_small.csv",
           encoding="ms949")

def minusToZero(inValue):
    if inValue < 0:
        inValue = 0
    return inValue

selloutDataS["QTY"] = \
    selloutDataS.QTY.apply(minusToZero)
```

그림 10. Dask 변환
Fig. 10. Dask Conversion

5. 실험 결과

5.1 Pandas 적용결과

Pandas 라이브러리만 사용하여 0.1GB에 모델을 구동한 결과 [그림 11]과 같이 약 1분10초가 소요된다.

```
finalResult.to_csv("./result_small.csv",
                   index=False, encoding="ms949")
print("completed")
elapsedtime = datetime.now() - starttime
print( 'time elapsed (hh:mm:ss.ms {})'.\
      format(elapsedtime))
```

```
data shape is (697996, 17)
completed
time elapsed (hh:mm:ss.ms 0:01:09.060388)
```

그림 11. 실험 결과 (Pandas only, 0.1GB)
Fig. 11. Test Result (Pandas only, 0.1GB)

1.5GB 데이터 적용 시 약 11분51초가 소요된다.

```
finalResult.to_csv("./result_large.csv",
                   index=False, encoding="ms949")
print("completed")
elapsedtime = datetime.now() - starttime
print( 'time elapsed (hh:mm:ss.ms {})'.\
      format(elapsedtime))
```

```
data shape is (10469940, 17)
completed
time elapsed (hh:mm:ss.ms 0:11:51.194153)
```

그림 12. 실험 결과 (Pandas only, 1.5GB)
Fig. 12. Test Result (Pandas only, 1.5GB)

5.2 Dask 적용결과

Pandas 라이브러리에 Dask를 적용하여 0.1GB에 모델을 구동한 결과 약 35초가 소요된다.

```
finalResult.to_csv("./result_small3.csv",
                   index=False, encoding="ms949")
print("completed")
elapsedtime = datetime.now() - starttime
print( 'time elapsed (hh:mm:ss.ms {})'.\
      format(elapsedtime))
```

```
completed
time elapsed (hh:mm:ss.ms 0:00:35.885249)
```

그림 13. 실험 결과 (Dask, 0.1GB)
Fig. 13. Test Result (Dask, 0.1GB)

1.5GB 데이터 적용 시 약 8분33초가 소요된다.

```
finalResult.to_csv("./result_large3.csv",
                   index=False, encoding="ms949")
print("completed")
elapsedtime = datetime.now() - starttime
print( 'time elapsed (hh:mm:ss.ms {})'.\
      format(elapsedtime))
```

```
completed
time elapsed (hh:mm:ss.ms 0:08:33.269202)
```

그림 14. 실험 결과 (Dask, 1.5GB)
Fig. 14. Test Result (Dask, 1.5GB)

데이터 크기에 따라 Pandas 와 Dask를 사용했을 때 [표 3]과 같이 성능을 요약할 수 있다.

표 3. 결과 비교
Table 3. Data Cleansing Performance

	Library	Performance	
		0.1GB	1.5GB
1	Pandas	70s	711s
2	Dask	35s	513s
summary		▽50%	▽28%

6. 결론

데이터 모델링 시 일반적으로 Pandas 라이브러리만 사용하여 분석한다. 하지만 신기술로 등장한 Dask를 활용한다면 고사양 노트북(2코어)만 활용해도 실험결과와 같이 약 40%의 성능 효과를 보여줌을 알수 있다. 파이썬이 단순히 테스트용 코드가 아닌 빅데이터 환경에서도 운영 가능한 언어가 될 수 있다. 기존 방법은 기가단위 데이터 이상인 경우 처리 속도가 급격히 저하되어 성능 이슈가 발생한다. 하지만 분산처리 가능한 Dask 라이브러리를 적용하면 Pandas의 기능을 향상된 속도에서 사용할 수 있다. 데이터 분석 시 가장 많은 시간을 투자하는 전처리 프로세스를 빠르게 처리하기 위해 Pyspark, Modin, Koala등 다양한 시도가 끊임없이 시도되고 있지만 성능을 높이기 위하여 기존 코드를 새로운 환경에서 다시 개발해야하는 어려움이 있다. 하지만 Dask는 파이썬 기존 개발코드를 최대한 유지하여 파이썬의 장점인 인공지능 관련 라이브러리가 많다는 점을 최대한 활용할 수 있다. 따라서 성능 향상을 위한 방법으로 Dask를 적용하는 방법을 논문에서 제시한다. 향후 추가로 연구할 부분은 Dask 와 Pandas의 함수 기능을 비교하여 기존 파이썬 코드와 Dask로 변환해야할 부분에 대해서 정확히 구분하여 튜닝 포인트를 정의할 필요가 있다. [8]

REFERENCES

[1] https://www.kaggle.com/kaggle/kaggle-survey-2018, 2018

[2] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica, "Spark:Cluster Computing with Working Set", Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, pp. 10-10, May 2010.

[3] Wes McKinney, "Python for Data Analysis", O'Reilly, pp. 52-58, Oct, 2017.

[4] Dask, https://dask.org, 2019

[5] Rafizul Islam M, Fahmida Kanij, "SPATIALITY, SEASONALITY AND INDEX ANALYSIS OF HEAVY METALS IN SOIL OF WASTE DISPOSAL SITE IN KHULNA OF BANGLADESH, Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, pp. 234-256, May 2017.

[6] Jason Brownlee, "Deep Learning for Time Series Forecasting: Predict the Future with NLPs, CNNs and LSTMs in Python", Machine Learning Mastery, pp. 161-164, Aug 2018.

[7] Lisandro D.Dalcin, Rodrigo R.Paz, Pablo A.Kler, "Parallel distributed computing using Python", Advances in Water Resources Volume 34, pp. 1124-1128, Sep 2011

[8] Micha Gorelick, Ian Ozsvald, "High Performance Python", O'Reilly, pp. 20-25, May 2020

저자약력

김 효 관(Hyo-Kwan Kim)

- 2001년 3월 ~ 2007년 8월: 성균관대학교 정보통신공학부 학사
- 2011년 9월 ~ 2013년 8월: 한국교통대학교 컴퓨터공학과 석사
- 2014년 3월 ~ 2017년 2월: 한국교통대학교 컴퓨터공학과 박사
- 2007년 3월 ~ 2017년 11월: 도담시스템스 소프트웨어개발 & 삼성 SDS 데이터분석그룹
- 2017년 12월 ~ 현재: 한국폴리텍대학 스마트금융과 교수

〈관심분야〉 인공지능, 금융데이터 분석, 핀테크

황 원 용(Won-Yong Hwang)

- 1997년 3월 ~ 2004년 2월: 고려대학교 전자 및 정보공학 학사
- 2009년 2월 ~ 2011년 2월: KAIST 소프트웨어대학원 공학석사 졸업
- 2004년 1월 ~ 2007년 2월: LG전자 MC연구소 휴대폰 솔루션 개발
- 2007년 3월 ~ 2008년 12월: LIG넥스원 지대공유도무기 개발
- 2011년 3월 ~ 2017년 5월: 삼성SDS 솔루션 개발팀(EFSS 팀)
- 2017년 6월 ~ 현재: 한국폴리텍대학 스마트금융과 교수

〈관심분야〉 인공지능, 블록체인, 핀테크