Michał Bernardelli[1]

Econometric modeling of panel data using parallel computing with Apache Spark

Summary
The aim of this article is to provide a method for determining the fixed effects estimators using MapReduce programming model implemented in Apache Spark. From many known algorithms two common approaches were exploited: the within transformation and least squares dummy variables method (LSDV). Efficiency of the computations was demonstrated by solving a specially crafted example for sample data. Based on theoretical analysis and computer experiments it can be stated that Apache Spark is an efficient tool for modeling panel data especially if it comes to Big Data.
Keywords: fixed effects estimator, panel data, Apache Spark, Big Data, MapReduce

1. Introduction

Nowadays, panel data are widely use in practice. Modeling and computations exploring the characteristics of such data, play a key role in modern economy and other fields of science. Main advantage behind this approach is the possibility to receive more accurate estimates of the model parameters and what follows much better assessment of the modelled phenomenon. In practice possibilities of getting information based on disaggregated data are often greater then in case of using even complex models, but on limited (aggregated) data sets. Unfortunately there are some cons of using panel data. Mostly there are much more data involved in the modeling. This means that the computations are costlier than those based only on times series or cross-sectional data. Also more sophisticated algorithms should be used to get the correct result. Available software often used in an econometric modeling, like Stata, SPSS or SAS, is already equipped with algorithms dedicated to panel data computations. However, those computer programs, usually with user-friendly interface, are

[1] Warsaw School of Economics, Collegium of Economic Analysis.

not so powerful concerning the effectiveness of calculations. It becomes noticeable when it comes to solving tasks with many variables and/or much volumes of input data.

Commonly concept of Big Data is identified with just a large volumes of data. Yet, in any existing definition, term Big Data is much wider than that. One of the popular definition is known by the name "3 V"[2] and characterized by attributes: volume, variety and velocity. Many authors extends this definition by adding two more attributes: variability and complexity. All the definitions have in common that processing Big Data is nontrivial task and there is a need to develop algorithms and solutions, which are able to deal with such enormous amount of data. In all modern alternatives the advantages of parallelization are exploited.

One of the most powerful frameworks for distributing computations on computer clusters, is Apache Spark. It is touted[3] as a fast and general engine for large-scale data processing. Since the date of creation (2009), the wide spectrum of algorithms arranged in libraries, were created. Among them there are implementations of the classical linear and logistic regression algorithms. What is missing, is the range of panel data methods.

The aim of this article is to provide a method for determining the parameters of the models based on panel data using MapReduce programming model implemented in Apache Spark. More precisely the fixed effects model was considered and two well-known methods of calculating parameter's estimates were presented, that is within transformation and least squares dummy variables approach. Efficiency of the computations was demonstrated by solving a specially crafted examples for sample data.

This paper is composed of five sections. The more detailed description of MapReduce model and Apache Spark framework is in section 2. Section 3 presents the brief theory behind the estimation methods of the panel data fixed effects model parameters. In section 4 there are fragments of codes of the algorithms and descriptions of the proposed methods. There are results of the computer numerical experiments as well. This paper ends with the summary in section 5.

[2] L. Doug, *3D Data Management: Controlling data volume, velocity, and variety*, 2001, http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D–Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf (retrieved 2016.04.05).

[3] http://spark.apache.org (retrieved 2016.04.05).

2. MapReduce and Apache Spark

MapReduce is a programming model for processing large volumes of data on computer clusters. It allows to perform computations in parallel or distributed way, while taking care of possible failures of individual nodes in the cluster. In fact it could be treated as a smart way of storing data and accessing it. MapReduce was developed by Google Inc., but nowadays more popular is the Open Source solution given by Apache (project Apache Hadoop[4]). The idea behind MapReduce model is simple: divide computationally challenging problem into sub problems that may be solve on a single machine (core). Of course not every problem can be parallelized, but many tasks or even parts of the bigger problem often may by calculated independently on separate machines.

The classic example, exploited extensively in the literature[5] showing the real power of the MapReduce approach, is to count the appearance of each word in a document. Without MapReduce the imposing solution is to sort the words in a document and count the same words located next to each other. With Big Data however it may be a problem if the file is larger than available memory. On the other hand suppose that there are 26 separate machines. We can split the set of words into disjoint 26 groups: each group of words beginning with different letter of an alphabet (assuming 26 letters in alphabet). Each group may be considered independently on a separate machine. Therefore in this case the speed-up related to the use of distributed hardware is (almost[6]) proportional to the number of available processors (cores, machines). MapReduce libraries have been written in many programming languages. Also different levels of optimization were used in various implementation of MapReduce concept. More information about potential of MapReduce can be found in Ullman[7].

Apache Spark could be treated as a new generation of MapReduce approach. Originally Spark was developed in 2009 at the University of California, Berkeley's AMPLab. Since 2010 is available as an Open Source software and since 2013 the Spark codebase was donated to the Apache Software Foundation becoming

[4] http://hadoop.apache.org (retrieved 2016.04.05).

[5] J. Dean, S. Ghemawat, *MapReduce: Simplified Data Processing on Large Clusters*, "Communication of ACM" 2008, vol. 51, issue 1, pp. 107–113.

[6] The time of splitting or sending data has to be considered. The real speed-up depends on the particular machine and the task.

[7] J.D. Ullman, *Designing good MapReduce algorithms*, "XRDS: Crossroads. The ACM Magazine for Students" 2012, vol. 19, pp. 30–34.

Apache Spark. Unlike MapReduce, Spark avoids serialization data to the hard disk so that in many applications the calculation works greatly faster. Spark is relatively faster than MapReduce, but calculations on Spark are more memory consuming then those on MapReduce. Depending on the task it could be perceived as an advantage or disadvantage.

As a proof of effectiveness of Spark it can be noted that in 2014 this software won so called Daytona Gray-Sort contest[8]. This competition involves sorting under specific rules 100 TB of data. Using the Spark technology they managed to break the record of MapReduce: three times faster with 10 times less machines (23 minutes vs. 72 minutes). There was also a successful attempt of sorting 1 PB[9] in 234 minutes.

There are also other aspects in favor of Spark. Three of them are definitely worth to mention. Firstly, Spark includes multiple additional – compared to MapReduce – useful operations, for example join or sample. Secondly, Spark supports many programming languages, including Java, which is the only one supported by MapReduce. These languages are Scala, R and Python. For all of them it is possible to use interactive shells. The third aspect of using Spark[10] is the availability of quite wide libraries: MLlib and ML packages[11]. MLlib is a machine learning library, with many implemented useful algorithms, including classification, regression, clustering, collaborative filtering, and dimensionality reduction. From econometric point of view there are different kind of regression implemented in those packages: linear and logistic, with or without regularization. However, none of available procedures can be used directly to an analysis of panel data. List of all functions with detailed descriptions could be found on the website with Apache Spark documentation[12].

All fragments of code in this article are written in Python[13] language. In every case of implementation the key role plays linear regression with ordinary least squares method, which is implemented in Apache Spark and can be found in the ML package.

[8] http://sortbenchmark.org (retrieved 2016.04.05).
[9] One petabyte is equal to 1024 terabyte.
[10] In version 1.6.1.
[11] In older versions there exists only MLib packge.
[12] http://spark.apache.org/docs/latest/mllib-guide.html (retrieved 2016.04.05).
[13] https://www.python.org (retrieved 2016.04.05).

3. Fixed effects estimator

Panel data econometrics is quite a new concept in the long history of econometric modeling. Classic of the literature in this field certainly include Arellando[14] and Hsiao[15], but there are also newer, worth mentioning monographs like Baltagi[16] or Diggle et al.[17] A typical model based on panel data could be presented in the form[18]

$$Y_{i,t} = \beta_0 + \beta_1 X_{1,i,t} + \beta_2 X_{2,i,t} + \ldots + \beta_k X_{k,i,t} + \alpha_i + \varepsilon_{i,t}, \qquad (1)$$

where:
$i = 1, 2, \ldots, n$ – individual's index,
$t = 1, 2, \ldots, T$ – time index,
$Y_{i,t}$ – dependent variable observed for individual i at time t,
$X_{k,i,t}$ – independent variable for individual i at time t,
β_k – parameter related to the k-th variable,
α_i – unobserved time-invariant individual effect,
$\varepsilon_{i,t}$ – error term for individual i at time t.
If the analyzed group of individuals is homogeneous, then it is possible to use an ordinary least squares method to calculate the estimators of the model parameters. Such an estimator is usually known as a pooled estimator, but it doesn't in fact exploit the presence of panels. For heterogeneous data, panels make possible to overcome a problem of bias. Depending on the data structure the fixed or random effects estimation may be used[19]. This article is devoted to the fixed effects model. In this model it is assumed that the unobserved effect α_i is the same for all observations for that individual. The detailed discussion concerning aspects of theoretical modeling of panel data and fixed effects estimation

[14] M. Arellano, *Panel Data Econometrics*, Oxford University Press, Oxford 2003.
[15] Ch. Hsiao, *Analysis of Panel Data*, Cambridge University Press, New York 2003.
[16] B. Baltagi, *Econometric Analysis of Panel Data*, Wiley, Chippenham 2013.
[17] P. Diggle, P. Heagerty, K.Y. Liang, S. Zeger, *Analysis of Longitudinal Data*, Oxford University Press, Oxford 2013.
[18] Sometimes the time dependent trend term is added to the equation.
[19] J.C. Gardiner, Z. Luo, L.A. Roman, *Fixed effects, random effects and GEE: What are the differences?*, "Statistics in Medicine" 2009, vol. 28, pp. 221–239.

may be found in Hsiao[20] or Wooldridge[21]. The key aspect of this article is the method of the estimation itself and its implementation in the Apache Spark framework. Therefore the theoretical properties of the estimators are omitted in favor of the description of the most known algorithms for calculating panel data model parameters.

At least three versions of the fixed effects approach are known in the literature and practice. The first uses within transformation, the second is based on the dummy variables, whereas the third is known by the name of first-differences method. From those methods the first two are presented in detail in this article. The third approach is conceptually close to the first one, but in fact it is rather even easier to implement. Therefore only two, basically different, approaches will be presented. Of course there are alternatives to those three listed methods, usually based on iterative algorithms, which could be even more computationally and memory efficient, but rather not so accurate[22].

In the first approach, so called within-groups method, mean values of the variables connected with observations in the given individual is calculated by averaging the observations for that individual:

$$\overline{Y_i} = \beta_0 + \sum_{j=1}^{k} \beta_j \bar{X}_{j,i} + \alpha_i + \bar{\varepsilon}_i, \tag{2}$$

where

$$\overline{Y_i} = \frac{1}{T}\sum_{t=1}^{T} Y_{i,t},$$

$$\bar{X}_{j,i} = \frac{1}{T}\sum_{t=1}^{T} X_{j,i,t},$$

$$\bar{\varepsilon}_i = \frac{1}{T}\sum_{t=1}^{T} \varepsilon_{i,t}.$$

[20] Ch. Hsiao, op.cit.

[21] J.M. Wooldridge, *Introductory Econometrics: A Modern Approach*, South-Western, Mason 2013.

[22] D. Tait, Ch.J. Cieszewski, I.E. Bella, *The stand dynamics of lodgepole pine*, "Canadian Journal Forest Research" 1986, vol. 18, pp. 1255–1260 or M. Strub, Ch.J. Cieszewski, *Base-age invariance properties of two techniques for estimating the parameters of site index models*, "Forest Science" 2006, vol. 52(2), pp. 182–186.

In equation (2) effect α_i seems to be not averaged, but because all observations for that individual are equal, then this term remains unaffected. As a result of subtraction equation (2) from equation (1) we get[23]

$$Y_{i,t} - \overline{Y_i} = \sum_{j=1}^{k} \beta_j \left(X_{j,i,t} - \bar{X}_{j,i} \right) + \varepsilon_{i,t} - \bar{\varepsilon}_i. \tag{3}$$

In this equation an unobserved effect α_i is absent and parameters β_j may be calculated for example using ordinary least squares method. It is worth to mention that also the intercept β_0 is drop out of the model, as well as any variable that remains constant (or slow changing) for each individual[24]. In the model described by equation (3) the variations about means of the dependent variable are explained in terms of the variations about the means of the independent variables connected with the group of observations in the given individual.

The second approach is called least squares dummy variable (LSDV) method. It involves adding a dummy variable for each individual. In this way the unobserved effect α_i is brought explicitly into the model. Assuming that there are exactly *n* panels we get

$$Y_{i,t} = \sum_{j=1}^{k} \beta_j X_{j,i,t} + \sum_{i=1}^{n} \alpha_i \delta_i + \varepsilon_{i,t}, \tag{4}$$

where δ_i is the binary variable, which takes the value 1 in case of an observation relating to the individual *i*. Analogously to the first approach, the model parameters after such a reformulation may be calculated using ordinary least squares method[25]. Also similar[26] is the absence of the intercept term β_0.

Two presented methods are mathematically equivalent. Although usually using the LSDV method is considered as not practical in case of the large number of individuals and connected with this the need of creating a large number of dummy variables. It is not recommended in such situation rather because of the high computer power and memory needed to solve this problem directly.

[23] In the literature one may encounter the same equation in the different notation

$$\ddot{Y}_{i,t} = \sum_{j=1}^{k} \beta_j \ddot{X}_{j,i,t} + \ddot{\varepsilon}_{i,t}.$$

[24] In these cases the fixed effects model is usually not the best choice.

[25] Frisch-Waugh theorem could be applied to LSDV estimator to take advantage of a special structure of matrices.

[26] Alternative is to include intercept at the expense of omission α_i for one individual.

Also in case of the non-balance data and large number of observations you can expect some numerical issues due to the pseudo-collinearity of binary variables[27]. It should be emphasized that it is possible to solve that kind of demanding tasks, but probably not by popular in econometric practice computer software. One alternative is indeed the Apache Spark that uses the parallel power of computer architecture. Nevertheless in most cases the first approach seems to be a far better choice, but still challenging in case of Big Data (especially if the problem is larger than the available memory of the computer).

In both presented methods a number of degrees of freedom (if the panel is balanced) is equal to $nT - (k+1) - n$, where k is the number of model (1) parameters, n is the number of individuals and T is the length of the time series. Therefore in some cases (e.g. short time series and a large number of variables) fixed effects estimator is not possible to obtain due to the not sufficient data. It is however not the case of this article, where we assume to deal with the completely opposite – Big Data – problems.

4. Implementation and numerical experiments

Main goal of this article was to present the possibility to solve Big Data problems of determining the panel data model coefficients in cases when classically used econometric software fails. Approach that was exploit as an alternative focuses on new, powerful open source framework released by Apache, which can be successfully used as an engine and the calculation tool. Spark's available libraries cover areas of regression and classification, but they still do not support properly the key methods in panel econometrics. This doesn't however mean that Apache Spark is useless. Contrary, using existing algorithms it is possible to get solutions of the basic panel data problems. In this article the fixed effects model and a way of calculating its parameters using Spark is described. The implementation of the presented two alternative methods, namely within transformation and least squares dummy variables approach, is using the strengths of the MapReduce concept described briefly in the second section.

[27] Solutions of numerical problems are not covered in this article, for more information about solving ill-conditioned or singular (due to the machine arithmetic precision) systems of linear equations see E. Cheney, D. Kincaid, *Numerical Mathematics and Computing*, Cengage Learning, Boston 2007.

This section is divided into three parts. In the first two some insight of the implementation of both of the methods is presented. In the third part there is the description of the construction of the test examples and the results of the computer simulations.

The first approach, described in the previous section – see equations (2)–(3) – uses within transformation. The proposed implementation can be divided into three steps:

1. calculate averages of observations for each individual,
2. transform variables (dependent and independent) using calculated averages and compute model parameters using ordinary least square method,
3. (optional) calculate error measurement or indicators of quality of the fit.

In those step, reading the input data is abandoned. However, it is assumed that data are kept in the DataFrame[28] variable named *parsedData* with the specific structure where first independent variable is the index of the individual.

Table 1. The first step of the within transformation approach

```
avgs = sorted(parsedData.select("label", "features")
            .map(lambda p: (p[1] [0],np.hstack((1, p[0],p[1] [1:]))))
            .reduceByKey(lambda a,b: a+b)
            .collect())
```

Source: own calculations.

The first step is a textbook example of use MapReduce approach. It also requires NumPy[29] – package for scientific computing. The whole step, due to the compact syntax of PySpark[30], may be written in one line only, which is instead presented in Table 1 in few lines just for clarity.

Second step of the implementation is presented in Table 2. Variable *avgs* was computed in the previous step and subsequent values of the vector *v* have the following interpretation: dependent value, the index of the individual, rest of independent variables. The second part of this step is the direct use of the functions from the library *pyspark.ml.regression* – in this case the implementation of the ordinary least squares method.

[28] See documentation at http://spark.apache.org/docs/latest (retrieved 2016.04.05).

[29] http://www.numpy.org (retrieved 2016.04.05).

[30] https://spark.apache.org/docs/latest/programming-guide.html (retrieved 2016.04.05).

Table 2. The second step of the within transformation approach

```
# row data transformation
LabeledPoint(
            v [0]-avgs[int(v [1])–1] [1] [1]/avgs[int(v[1])–1] [1] [0],
            v [2:]-avgs [int (v [1]) – 1] [1] [2:]/avgs [int (v [1]) – 1] [1] [0])
```

```
# build the model
lr = LinearRegression(maxIter=100, fitIntercept=False)
model = lr.fit (parsedData)
print("Coefficients: " + str (model.coefficients))
print("Intercept: " + str (model.intercept))
```

Source: own calculations.

The third step is optional, but rather useful. It allows to evaluate the model on test data or calculated selected measurements of the quality of the model. In Table 3 there is an implementation of mean squared error (MSE), which allows to fully benefit from the potential of Apache Spark and MapReduce.

Table 3. The third step of the within transformation approach

```
# Evaluate the model on training data
MSE = model.transform(parsedData)
            .map(lambda p: (p['label']-p['prediction'])**2)
            .reduce(lambda x, y: x+y) / parsedData.count()
print("Mean Squared Error = " + str(MSE))
```

Source: own calculations.

The second part of this section is devoted to the least squares dummy variables approach – equation (4) from the previous section. The proposed implementation can be presented in two steps.

1. Creating dummy variables and compute model parameters using ordinary least square method,
2. (optional) calculate error measurement or indicators of quality of the fit.

The second step is exactly the same as the third step of the within transformation approach (see Table 3), whereas the first step is similar to the second step of that method (see Table 2). In Table 4 there is an implementation of key aspects of this step. It needs to be highlighted that building the model in two presented approaches differs by the parameter *fitIntercept*. In within transformation there is no intercept, but in LSDV approach an intercept is necessary.

Table 4. The first step of the least squares dummy variables approach

```
# row data transformation (n – number of individuals)
v2 = [0] * n
v2[int(v[2])] = 1
LabeledPoint(v[0], v[2:] +v2)
```

```
# build the model
lr = LinearRegression (maxIter=100, fitIntercept=True)
model = lr.fit (parsedData)
print ("Coefficients: " + str (model.coefficients))
print ("Intercept: " + str (model.intercept))
```

Source: own calculations.

The third part of this section contains some practical results of the test examples. Due to the limited space only the most demanding computationally executed test is presented. For the testing purpose model with $k = 100$ variables was considered, where the number of individuals equal $n = 100$. Training data set consists of randomly chosen million observations, but the individual error and model parameters were fixed. Size of the training (and test at the same time) data was almost two gigabytes, which is rather an abstractly large, for commonly econometric software, amount of data[31]. Three methods were compared:

1. pooled estimator, which doesn't pull benefits from the panels structure of data,
2. within transformation method,
3. least squares dummy variables method.

Calculations on a single, five-core machine in each case lasted approximately 8 minutes. It should be stated that using more executors will improve the speed of calculations on more then a local machine. Also, due to the excellent scaling properties of Spark, there is a possibility to solve much larger then two gigabytes problems.

Results of the numerical experiment are gathered in Table 5. Mean Squared Error calculated for the whole training data set is presented in Table 6. There are two things, which should be point out. Firstly parameter's estimates in pooled estimator are worst then those in fixed effects estimators (MSE 1.2 vs 0.0). Secondly within transformation and least squares dummy variables approach gives in fact the same estimates, but values of the parameters listed in Table 5 are different, because the within transformation is one of the methods that shifts variables with arithmetic means and therefore parameters β_j in equations (3) and (4) have different meanings.

[31] Data may be stored in the database (e.g. Hive) table instead of the text file.

Table 5. Results of the numerical experiment. Values of the selected model parameters (rounded to 4 decimal places)

Param.	Polled estim.	Within trans.	LSDV	Param.	Polled estim.	Within trans.	LSDV
Inter.	11.8334	0.0	15.8438	β_{90}	–10.9995	–11.0	–5.1011
β_1	–4.0009	–4.0	–4.0	β_{91}	–4.0000	–4.0	–3.5043
β_2	8.9973	9.0	9.0	β_{92}	9.0027	9.0	–4.3057
β_3	–15.9968	–16.0	–16.0	β_{93}	–15.9995	–16.0	–5.2668
β_4	25.0018	25.0	25.0	β_{94}	25.0012	25.0	–3.1797
β_5	–35.9988	–36.0	–36.0	β_{95}	–36.0018	–36.0	–2.7809
β_6	49.0031	49.0	49.0	β_{96}	48.9977	49.0	–2.4281
β_7	–64.0012	–64.0	–64.0	β_{97}	–63.9976	–64.0	–5.2126
β_8	81.0017	81.0	81.0	β_{98}	80.9966	81.0	–2.0347
β_9	*99.9976*	100.0	100.0	β_{99}	99.9999	100.0	–4.6905
β_{10}	–11.0005	–11.0	–11.0	β_{100}	–11.0013	–11.0	–3.6860

Source: own calculations.

Table 6. Results of the numerical experiment. Mean Squared Error

Polled estimator	Within transformation	Least squares dummy variables
1.18470720786	1.6155932558e-25	1.6155932558e-25

Source: own calculations.

It is worth to mention that volume of data used in numerical example is still quite a small task for a standard laptop computer. Real life examples could have billions of observations. This however gives the basis for the conviction of the high efficiency of the proposed solution.

5. Conclusions

In this article the proposition of extending the Apache Spark libraries into new functions dedicated to the calculating fixed effects model estimators. Presented two equivalent methods that is within transformation and least squares dummy variables approach, can be used to modeling Big Data with panel characteristics. Based on theoretical analysis and performed computer experiments it is justified to draw the following conclusions:

1. Within transformation approach is computationally better than LSDV method and should be used in case of large number of individuals.
2. Apache Spark can be successfully used in panel data modeling.
3. MapReduce concept of parallelization allows to deal with Big Data problems.

Implementations described in this article should be however treated as a proof of suitability of Apache Spark framework. There are many other algorithms, which should be implemented if this software is to be used by a wider group of people. From the things from the top of the list probably it is worth to mention:

- random effects estimator,
- statistical tests like Breusch-Pagan, Pesaran, Friedman, Frees, etc.
- different than MSE measures of errors.

References

Arellano M., *Panel Data Econometrics*, Oxford University Press, Oxford 2003.

Baltagi B., *Econometric Analysis of Panel Data*, Wiley, Chippenham 2013.

Cheney E., Kincaid D., *Numerical Mathematics and Computing*, Cengage Learning, Boston 2007.

Dean J., Ghemawat S., *MapReduce: Simplified Data Processing on Large Clusters*, "Communication of ACM" 2008, vol. 51, issue 1, pp. 107–113.

Diggle P., Heagerty P., Liang K.Y., Zeger S., *Analysis of Longitudinal Data*, Oxford University Press, Oxford 2013.

Gardiner J.C., Luo Z., Roman L.A., *Fixed effects, random effects and GEE: What are the differences?*, "Statistics in Medicine" 2009, vol. 28, pp. 221–239.

Hsiao Ch., *Analysis of Panel Data*, Cambridge University Press, New York 2003.

Strub M., Cieszewski Ch.J., *Base-age invariance properties of two techniques for estimating the parameters of site index models*, "Forest Science" 2006, vol. 52(2), pp. 182–186.

Tait D., Cieszewski Ch.J., Bella I.E., *The stand dynamics of lodgepole pine*, "Canadian Journal Forest Research" 1986, vol. 18, pp. 1255–1260.

Ullman J.D., *Designing good MapReduce algorithms*, "XRDS: Crossroads. The ACM Magazine for Students" 2012, vol. 19, pp. 30–34.

Wooldridge J.M., *Introductory Econometrics: A Modern Approach*, South-Western, Mason 2013.

Internet sources

Doug L., *3D Data Management: Controlling data volume, velocity, and variety*, 2001, http://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D–Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf (retrieved 2016.04.05).

http://hadoop.apache.org (retrieved 2016.04.05).

http://sortbenchmark.org (retrieved 2016.04.05).

http://spark.apache.org (retrieved 2016.04.05).

http://www.numpy.org (retrieved 2016.04.05).

https://www.python.org (retrieved 2016.04.05)

* * *

Ekonometryczne modelowanie danych panelowych z wykorzystaniem obliczeń równoległych na Apache Spark

Streszczenie

Celem artykułu jest przedstawienie sposobu wyznaczania estymatora *fixed effects* przy użyciu modelu programowania MapReduce zaimplementowanego w Apache Spark. Spośród wielu znanych algorytmów zostały wykorzystane dwa popularne podejścia: transformacja *within* oraz *least squares dummy variables method* (LSDV). Efektywność obliczeń wykazano, rozwiązując specjalnie spreparowany przykład dla wygenerowanej losowo próbki danych. Na podstawie analizy teoretycznej i eksperymentów numerycznych można stwierdzić, że Apache Spark jest efektywnym narzędziem do modelowania danych panelowych, zwłaszcza jeśli chodzi o Big Data.

Słowa kluczowe: estymator fixed effects, dane panelowa, Apache Spark, Big Data, MapReduce