CSCI 654-01—Foundations of Parallel Computing Programming Project 4

Prof. Alan Kaminsky—Fall Semester 2018
Rochester Institute of Technology—Department of Computer Science

---

Overview

Write a map-reduce parallel program using the Parallel Java Map Reduce (PJMR) framework in the Parallel Java 2 Library. Run the program on a cluster parallel computer to learn about big data and map-reduce parallel programming.

Help with your project: I am willing to help you with the design of your project. I am willing to help you debug your project if the code isn't working. However, for help with design or debugging issues *you must come see me in person.* Either visit me during office hours or make an appointment. I will not help you with design or debugging issues via email. If it's the evening of the project deadline and I have gone home, you are on your own. Plan and work ahead so there will be plenty of time for me to help you if necessary.

U.S. Census Diversity Index

The United States Census Bureau (USCB) estimates the number of persons in each county in each state, categorized by age, gender, race, and other factors. USCB uses six racial categories: White; Black or African American; American Indian or Alaska Native; Asian; Native Hawaiian or Other Pacific Islander; Two or more races.

For example, here are the USCB's population estimates for Monroe County, New York, as of July 1, 2017:

White	573928
Black or African American	121423
American Indian or Alaska Native	3074
Asian	29053
Native Hawaiian or Other Pacific Islander	517
Two or more races	19667
Total	747662

The **diversity index** *D* for a population is the probability that two randomly chosen individuals in that population will be of different races. The diversity index is calculated with this formula, where N_i is the number of individuals in racial category *i* and *T* is the total number of individuals:

$$D = 1/T^2 \sum_{i=1}^{6} N_i (T - N_i)$$

For example, the diversity index of Monroe County, New York as of July 1, 2017 is 0.38216. The diversity index should be calculated using double precision floating point (type `double`).

I have downloaded a census dataset from the USCB web site. The dataset contains information for every county in every state from 2010 to 2017. I installed the full dataset on the `tardis` machine in a file named `/var/tmp/ark/USCensus/cc-est2017-alldata.csv`. You can log into `tardis` and look at the dataset. I also partitioned the dataset and installed the partitions on the local hard disks of the `dr00` through `dr09` nodes of the `tardis` cluster. Each node has one file named `/var/tmp/ark/USCensus/cc-est2017-alldata.csv`. Each file totals about 14 megabytes, and the whole dataset totals about 140 megabytes.

Each record in the dataset (each line in the file) consists of several fields. Each field is separated from the next field by a comma. The fields contain the following information. Fields are indexed from left to right starting at 0. Only the fields needed for this project are listed.

Index	*Name*	*Contents*	*Type*
3	STNAME	State name	String
4	CTYNAME	County name	String
5	YEAR	Year	Integer
6	AGEGRP	Age group	Integer
10	WA_MALE	White male population	Integer
11	WA_FEMALE	White female population	Integer
12	BA_MALE	Black or African American male population	Integer
13	BA_FEMALE	Black or African American female population	Integer
14	IA_MALE	American Indian or Alaska Native male population	Integer
15	IA_FEMALE	American Indian or Alaska Native female population	Integer
16	AA_MALE	Asian male population	Integer
17	AA_FEMALE	Asian female population	Integer

18	NA_MALE	Native Hawaiian or Other Pacific Islander male population	Integer
19	NA_FEMALE	Native Hawaiian or Other Pacific Islander female population	Integer
20	TOM_MALE	Two or more races male population	Integer
21	TOM_FEMALE	Two or more races female population	Integer

The YEAR field is an integer giving the record's year:
1 = April 1, 2010 census population
2 = April 1, 2010 population estimates base
3 = July 1, 2010 population estimate
4 = July 1, 2011 population estimate
5 = July 1, 2012 population estimate
6 = July 1, 2013 population estimate
7 = July 1, 2014 population estimate
8 = July 1, 2015 population estimate
9 = July 1, 2016 population estimate
10 = July 1, 2017 population estimate

The AGEGRP field is an integer giving the record's age group. We are only interested in records with AGEGRP = 0; such records contain the total population for all age groups.

The number of individuals in a racial category is the sum of the number of males and the number of females. For example, the number of white individuals is WA_MALE + WA_FEMALE.

The USCB web site has a document that describes the census dataset in full detail.

You will write a Parallel Java Map Reduce (PJMR) job to analyze the census data and calculate the diversity index for every county in a given state or states for a given year. The program will also calculate the diversity index for the overall population of each state.

Program Input and Output

The census data analysis program's command line arguments are a list of comma-separated node names, the name of the file containing the census data, the year (an integer from 1 to 10), and zero or more state names. (The state names might include spaces and so must be enclosed in quotation marks, such as `"New York"`.) If there are no state names, the program must analyze all counties in all states for the given year. If there are one or more state names, the program must analyze all counties in the given states for the given year. The number of mapper threads per mapper task is also specified using the `pj2` program's `"threads="` option.

The census data analysis program's output consists of a series of sections. Each section reports results for one state. The sections appear in ascending order of the state name.

Each section consists of a series of lines. The first line consists of the state name, two tab characters, the state's overall diversity index calculated from the total population of that state, and a newline. The second and subsequent lines each consist of a tab character, the county name, a tab character, the county's diversity index calculated from the population of just that county, and a newline. The county lines are printed in descending order of the diversity index. If multiple county lines have the same diversity index, those county lines are printed in ascending order of the county name.

You must use the following code to print the first line in each section:

```
    System.out.printf ("%s\t\t%.5g%n", stateName, diversityIndex);
```

You must use the following code to print the second and subsequent lines in each section:

```
    System.out.printf ("\t%s\t%.5g%n", countyName, diversityIndex);
```

A PJMR job is a cluster parallel program. This means that, when running on the `tardis` cluster, you must include the `"jar="` option on the `pj2` command line. The JAR file must contain all the Java class files (`.class` files) in your project.

Here is an example of the analysis program running on the `tardis` cluster. (*Note:* The lines reporting job statistics were printed by `pj2`; they are not part of the analysis program's output.)

```
$ java pj2 jar=p4.jar DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 10 "New York" "Delaware"
Job 159 launched Mon Nov 19 09:54:44 EST 2018
Job 159 started Mon Nov 19 09:54:44 EST 2018
Delaware                0.45971
        New Castle County       0.50210
        Kent County     0.48199
        Sussex County   0.29902
New York                0.47584
        Queens County   0.65390
        Kings County    0.62253
        Bronx County    0.60372
        New York County 0.53530
        Westchester County      0.42755
        Nassau County   0.41871
        Richmond County 0.40180
        Albany County   0.39836
        Monroe County   0.38216
        Rockland County 0.37691
        Schenectady County      0.36599
        Erie County     0.34747
        Onondaga County 0.34385
        Tompkins County 0.33426
        Orange County   0.33248
        Dutchess County 0.32254
        Franklin County 0.28961
        Suffolk County  0.27567
        Sullivan County 0.26730
```

```
      Oneida County  0.25274
      Broome County  0.25194
      Rensselaer County      0.24370
      Jefferson County       0.23035
      Ulster County  0.22939
      Niagara County 0.22235
      Chemung County 0.21365
      Greene County  0.18810
      Orleans County 0.18458
      Columbia County0.18228
      Seneca County  0.15810
      Putnam County  0.15491
      Clinton County 0.15324
      Cattaraugus County     0.15288
      Wyoming County 0.14762
      Cayuga County  0.14600
      Genesee County 0.13357
      Saratoga County0.13325
      Montgomery County      0.12806
      Livingston County      0.12422
      Wayne County   0.12255
      Chautauqua County      0.12089
      St. Lawrence County    0.12029
      Ontario County 0.12029
      Essex County   0.11883
      Otsego County  0.11324
      Washington County      0.10925
      Cortland County0.10132
      Madison County 0.10068
      Steuben County 0.096877
      Delaware County0.091645
      Fulton County  0.091441
      Allegany County0.085892
      Warren County  0.082873
      Schoharie County       0.081710
      Oswego County  0.075486
      Herkimer County0.073554
      Hamilton County0.069286
      Chenango County0.067883
      Schuyler County0.067763
      Tioga County   0.067176
      Yates County   0.060257
      Lewis County   0.057358
Job 159 finished Mon Nov 19 09:54:46 EST 2018 time 1689 msec
```

Software Requirements

1. The program must be run by typing this command line on the `tardis` cluster:
2. `java pj2 jar=<jar> threads=<NT> DivIndex <nodes> <file> <year> [ "<state>" ... ]`
    - `<jar>` is the name of the JAR file containing all of the program's Java class files.
    - `<NT>` is the number of mapper threads per mapper task; if omitted, the default is 1.

- `<nodes>` is a comma-separated list of cluster node names on which to run the analysis. One or more node names must be specified.
- `<file>` is the name of the file on each node's local hard disk containing the census data to be analyzed.
- `<year>` is the year to be analyzed. It must be an integer from 1 to 10.
- `<state>` is a state name to be analyzed. There can be zero or more state names. A particular state name must not appear more than once.

Note: This means that the program's class must be named `DivIndex`, this class must not be in a package, and this class must extend class edu.rit.pjmr.PjmrJob.

3. If the command line does not have the required number of arguments, if any argument is erroneous, or if any other error occurs, the program must print an error message on the console and must exit. The error message must describe what the problem is. The wording of the error message is up to you.
4. The program must print a series of lines on the console as specified above under "Program Input and Output."

   *Note: If your program's output does not conform **exactly** to Software Requirements 1 through 3, **you will lose credit** on your project.* See the Grading Criteria below.

Software Design Criteria

1. The program must be a map-reduce job using the Parallel Java Map Reduce framework.
2. The program must be designed using object oriented design principles as appropriate.
3. The program must make use of reusable software components as appropriate.
4. Each class or interface must include a Javadoc comment describing the overall class or interface.
5. Each constructor and method within each class or interface must include a Javadoc comment describing the overall constructor or method, the arguments if any, the return value if any, and the exceptions thrown if any.

   Note: See my Java source files which we studied in class for the style of Javadoc comments I'm looking for.

   *Note: If your program's design does not conform to Software Design Criteria 1 through 5, **you will lose credit** on your project.* See the Grading Criteria below.

Submission Requirements

Your project submission will consist of a ZIP file containing the Java source file for every class and interface in your project. Put all the source files into a ZIP file named "`<username>.zip`", replacing `<username>` with the user name from your Computer Science Department account. On a Linux system, the command for creating a ZIP file is:

```
zip <username>.zip *.java
```

DO NOT include the Parallel Java 2 Library in your ZIP file!

Send your ZIP file to me by email at ark@cs.rit.edu. Include your full name and your computer account name in the email message, and include the ZIP file as an attachment.

When I get your email message, I will extract the contents of your ZIP file into a directory. I will set my Java class path to include the directory where I extracted your files, plus the Parallel Java 2 Library. I will compile all the Java source files in your program using the JDK 1.7 compiler. I will then send you a reply message acknowledging I received your project and stating whether I was able to compile all the source files. If you have not received a reply within one business day (i.e., not counting weekends), please contact me. Your project is not successfully submitted until I have sent you an acknowledgment stating I was able to compile all the source files.

The submission deadline is Monday, December 3, 2018 at 11:59pm. The date/time at which your email message arrives in my inbox will determine whether your project meets the deadline.

You may submit your project multiple times up until the deadline. I will keep and grade only the most recent successful submission. There is no penalty for multiple submissions.

If you submit your project before the deadline, but I do not accept it (e.g. I can't compile all the source files), and you cannot or do not submit your project again before the deadline, the project will be late (see below). ***I STRONGLY advise you to submit the project several days BEFORE the deadline, so there will be time to deal with any problems that might arise in the submission process.***

Grading Criteria

I will grade your project by:

- (10 points) Evaluating the design of your program, as documented in the Javadoc and as implemented in the source code.
  - All of the Software Design Criteria are fully met: 10 points.
  - Some of the Software Design Criteria are not fully met: 0 points.
- (20 points) Running your map-reduce parallel program. There will be twenty test cases, each worth 1 point. For each test case, if the program runs using the command line in Requirement 1 and the program produces the correct output, the test case will get 1 point,

otherwise the test case will get 0 points. "Correct output" means "output fulfills *all* the Software Requirements *exactly*."
- (30 points) Total.

When I run your programs, the Java class path will point to the directory with your compiled class files, plus the Parallel Java 2 Library. I will use JDK 1.8 to run your program.

I will grade the test cases based *solely* on whether your program produces the correct output as specified in the above Software Requirements. *Any* deviation from the requirements will result in a grade of 0 for the test case. This includes errors in the formatting (such as extra spaces), output lines not terminated with a newline, and extraneous output not called for in the requirements. The requirements state exactly what the output is supposed to be, and there is no excuse for outputting anything different. If any requirement is unclear, please ask for clarification.

If there is a defect in your program and that same defect causes multiple test cases to fail, I will deduct points for *every* failing test case. The number of points deducted does *not* depend on the size of the defect; I will deduct the same number of points whether the defect is 1 line, 10 lines, 100 lines, or whatever.

After grading your project I will put your grade and any comments I have in your encrypted grade file. For further information, see the Course Grading and Policies and the Encrypted Grades.

Test Cases

The grading test case commands were as follows. The correct outputs are in the files `out01.txt` through `out20.txt` in this ZIP archive. For test cases 1–3, the wording of the error message was up to you, as long as it described the problem.

IMPORTANT NOTE
Running on the `tardis` cluster, each test case 4–20 should have taken about 4 seconds. I set a 20-second timeout on each test case. If the program timed out, it indicates a bad design.

IMPORTANT NOTE
I have accurately recorded in your grade file what I observed during my testing. If you run the test cases and do not get the same results as I, then there is a flaw in your design somewhere that causes inconsistent results (perhaps due to incorrect parallelization), and I am not going to change your grade for the original submission. You may still resubmit your project.

```
1.  java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv
2.  java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv one "Delaware"
3.  java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 1 "Delaware" "Delaware"
```

```
4.  java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 1 "Alabama"
5.  java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 2 "Arkansas"
6.  java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 3 "Arizona"
7.  java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 4 "Arizona"
8.  java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 5 "Colorado" "California"
9.  java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 6 "Delaware" "Connecticut"
10. java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 7 "Florida" "Georgia"
11. java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 8 "Idaho" "Hawaii"
12. java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 9 "Illinois" "Indiana"
13. java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 10 "Iowa" "Kansas"
14. java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 1 "Kentucky" "Louisiana"
15. java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 2 "Massachusetts" "Maryland" "Maine"
16. java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 3 "New Mexico" "New Jersey" "New Hampshire"
17. java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 4 "New York" "North Carolina" "North Dakota"
18. java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 5 "Ohio" "Pennsylvania" "Oregon"
19. java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 6 "Rhode Island" "South Dakota" "Tennessee"
20. java pj2 debug=none timelimit=20 jar=p4.jar threads=2 DivIndex
dr00,dr01,dr02,dr03,dr04,dr05,dr06,dr07,dr08,dr09 /var/tmp/ark/USCensus/cc-
est2017-alldata.csv 7
```

Late Projects

If I have not received a successful submission of your project by the deadline, your project will be late and will receive a grade of zero. You may request an extension for the project. There is no penalty for an extension. See the Course Policies for my policy on extensions.

Plagiarism

Programming Project 4 must be entirely your own individual work. I will not tolerate plagiarism. If in my judgment the project is not entirely your own work, you will automatically receive, as a minimum, a grade of zero for the assignment. See the Course Policies for my policy on plagiarism.

Resubmission

If you so choose, you may submit a revised version of your project after you have received the grade for the original version. However, if the original project was not successfully submitted by the (possibly extended) deadline or was not entirely your own work (i.e., plagiarized), you are not allowed to submit a revised version. Submit the revised version via email in the same way as the original version. I will accept a resubmission up until 11:59pm Wednesday 12-Dec-2018. You may resubmit your project multiple times up until the deadline; I will keep and grade only the most recent successful resubmission; there is no penalty for multiple resubmissions. I will grade the revised version using the same criteria as the original version, then I will subtract 3 points (10% of the maximum possible points) as a resubmission penalty. The revised grade will replace the original grade, even if the revised grade is less than the original grade.