CS 6240: Project Final Report

**Goals**: (1) Report the results of your project. (2) Practice an important job skill: presenting your work to others in written form.

Each team has to create all deliverables from scratch. In particular, it is not allowed to copy another team’s code or text and modify it. If you use publicly available code or text, you need to **cite the source** in your report!

All deliverables for this HW are due as stated on Blackboard. For late submissions you will lose one percentage point per hour after the deadline. This HW is worth 100 points and accounts for 30% of your overall homework score. Please submit your solution through the Blackboard assignment submission system and make sure your report is a **PDF** file. All programs need to be submitted as source code files, e.g., MyProgram.java. Package everything into a single ZIP file. Make sure you are using standard ZIP, i.e., the format also used by Windows archives.

**File naming convention**: use the last names of all team members, sorted alphabetically, as the file name. For instance, the two-person team Joe Smith and Mary Miller would use file name MillerSmith.zip for the report and MillerSmith.pdf for the presentation.

# Project Final Report

The report is your opportunity to show us that you mastered MapReduce and are now able to solve realistic problems. Make sure that it properly reflects the effort and thoughts you put into this project. When preparing your project report, use the following structure:

**Introduction**. Start with a brief summary of your project topic and analysis goals. This is similar to the summary you wrote for the project proposal, but should be updated, especially if your goals changed. Also include a brief summary of the highlights of your project, e.g., if you found a clever way to solve a difficult problem or discovered something interesting in the data. Typically the entire introduction should not be longer than 1 page in the format of this document.

**Data**. As for the progress report, briefly describe the data you used, where you got them from, and if you had to do anything special to make them available for your project. (For example, if you had to collect data in real-time from Twitter, briefly explain how you did this. Or if you did something clever to get a big data set uploaded to S3, briefly describe your approach. This will count as a helper task.) Show a few lines of each input file.

**Technical discussion**. Discuss technical details how you solved the problem. For each task, do the following:

* Briefly discuss the purpose of the task.
* Explain in a few sentences the main idea for solving the problem, including any design patterns you considered and/or used, and show the pseudo-code. Do NOT just copy and paste source code into your report. (Exceptions are SQL, HiveQL, and Pig Latin, which by design are concise and hence can be included directly if appropriate.) We want to be able to quickly understand the main points of the algorithm without having to go through pages of boilerplate code. Look at the lecture slides and notes for pseudo-code examples. If one of your programs’ pseudo-code is (almost) identical to code that appears on a lecture slide, just reference the slide.
* Mention which of the files in your submitted ZIP file contains the corresponding source code.
* Discuss concrete results. In particular, show the appropriate graphs and tables to present analytical results or performance numbers. Always explain on how many machines of what type your program ran and how much time it took.
  + For example, assume you computed PageRank for a sample of the Web crawl. Then you should (1) discuss the relevant properties of the input data (how many nodes and edges the sample had, how many gigabytes, if it was a graph with longer paths or not), (2) report important performance numbers (e.g., a graph showing how long each iteration of PageRank took on 5 versus 10 versus 20 small machines; a graph showing how the PageRank of a popular page changed in each iteration and so on), and (3) briefly discuss the presented results, e.g., if anything looks interesting or surprising.
* Make sure you discuss the “clever solution” or analysis that goes beyond the “reasonable but straightforward” way of doing things (see initial project assignment for more details). Mention concrete numbers, e.g., from log files, that you used to decide if your program is efficient and scalable or not. Mention which of the log files etc in your submitted ZIP file contains the corresponding detailed numbers you used to evaluate your programs.

**Setup challenges**. This most likely only applies to teams who worked with Hive or Weka. Even though using these tools is *conceptually* straightforward, we know how challenging it can be to get them to work in *practice* due to incomplete and inaccurate documentation and the fact that the entire Hadoop universe is rapidly evolving. Significant effort required to solve such setup issues will be credited as a project contribution *if you do the following*:

* Discuss briefly what the main challenges were, e.g., to get Hive to work on EMR.
* Explain your solution is such a way that we can reproduce it: What parameters or settings did you use? What did you have to select in the EMR interface? What did you have to do in your program code to make it work? Did you have to copy data around in some clever way? Etc.
* Do not add “fluff”, i.e., obvious things or trivialities. Focus on the real setup-related challenges. Examples of fluff that should **not** be included here: your program did not run because you had a typo in the S3 path or you wrote an incorrect Hive query.
* Provide convincing evidence that your Hive job or data mining job using Weka indeed worked on Amazon EMR, not just on your local machine. E.g., include a screenshot in the report and include a corresponding EMR log file or similar in the submitted ZIP file.

**Conclusion**. Briefly summarize your main results and contributions and point out possible extensions or alternatives that might be worth exploring in future work. The conclusions should be about 0.5 to 1 page in this document’s format.

Additional recommendations: Include a brief summary of the major problems you faced and how you solved them. This way we can better evaluate the work you put into this project, especially if you attempted a task different from (and hence more risky than) the algorithms we discussed in class. For instance, assume you wrote MapReduce code to train a decision tree in parallel. Since we did not discuss this in detail in the course, you probably ran into non-trivial challenges when trying to come up with a good way of parallelizing it. Briefly explain where you got stuck and what you tried to get around the problem. However, do not try to “pad” your report with trivial results. For instance, if you created 10 obvious minor variations of some code, then just explain the main version and very briefly state how the others were different. Do not re-explain the same thing 10 times just to fill up the report.

Grading will be based on the quality and amount of work you put into the project, the use of non-trivial algorithms based on a careful analysis of their performance and scalability, and the quality of the presentation in the report. In particular, reports of high quality present the most important points clearly and compactly, without adding unnecessary fluff or lengthy discussions of minor or obvious aspects.

# Deliverables

1. The report as discussed above. (1 PDF file)
2. Your source code. (multiple .java or Pig or HiveQL files, scripts you used etc.)
3. Carefully selected log files and similar that you used to determine if your programs are efficient or not. Complete reports cite the relevant numbers (e.g., amount of data sent from Mappers to Reducers) in the report and then include the corresponding log file for reference in the ZIP file.
4. Optional: interesting and sufficiently small result files that you want to share or reference in your report. (plain text ASCII files, JPG or similar)