Uni Freiburg, Web Science Group
Prof. Peter Fischer
Systems Infrastructure for Data Science - Winter 2014/15

Exercise Sheet #11: Map/Reduce: Joins, Iterative Workloads

January 22, 2015

1 Map/Reduce Joins

Map/Reduce does provide a direct implementation of the join operator. Instead, multiple approaches have been proposed to express joins in an efficient manner.

A. The most obvious choice would be a *reduce-side* join, where the map stage just prepares data and reduce performs the actual joins. Describe map and reduce funtions (in your own words or using code) and describe how the join in computed.

B. Would it also be possible to perform a job purely on the map side? What conditions would have to be fulfilled, and how would the map and reduce funtions look like?

C. What would be the respective benefits and disadvantages of *reduce-side* and *map-side* joins?

2 Social Networks and Map/Reduce

In previous exercise sheet, we have studied how to determine common friends of users A and B on social graph. The assumption has been that these users A and B are already friends. To understand the limitations of Map/Reduce based approaches (including Pig), let us consider the following computations:

A. Compute common friends of users that are not directly connected (e.g., A is friend with C, B is friend with C, but A and B are not friends)

B. Determine the set of friends that are also connected by an intermediary friend (e.g. A is friend with B, B is friend with C; result includes B and C)

C. Is it possible to compute (with Map-Reduce) if two arbitrary users are connected, i.e. they have a path of friends between them? How would you approach this problem?