A Comprehensive Reduction Java Library for Java

Mostafa Mehrabi, Oliver Sinnen, Nasser Giacaman
mmeh012@aucklanduni.ac.nz, o.sinnen, n.giacaman{@auckland.ac.nz}
The University of Auckland – Engineering Depratment
New Zealand

November 14, 2014

Abstract

Reduction in computer science is the process of combining two or more elements into one. This process is widely used by network based applications for integrating results from different computers of a network. It also seems reasonable to use the same mechanism in shared memory applications that run on a single computer. That is, reducing the results that are obtained from different processors of a computer into the final result. However, there are not many libraries that facilitate reduction on single computers, as the main focus has been on network based applications thus far.

Considering the benefits of reduction for improving performance on shared memory applications, developing assistant libraries in this scope is quite worth while. In this paper we have introduced an extensive reduction library that was developed under Java. The object oriented considerations of the design have been explained, and it has been clarified how users benefit from them. Moreover, we have compared the features provided by our design with a few others that are available in this field. Further examples in this paper help with clearer understanding of the logic of our design.

Key words: Reduction, ParallelIterator, Shared Memory

1 Introduction

Fast technological growth has improved the accuracy of computations substantially due to the large amount of detailed data that we are able to collect now. Subsequently, the calculations that are performed on data are now more expensive in terms of resources. That is, there are considerably more resources required for performing computations than it used to be. Therefore, speed of computation (i.e. performance) is limited by the boundaries set by our resources. On a single computer, the number of CPU cycles can only be increased up to a certain extent due to physical limitations. Because of the problem thereof, many applications are moving on cloud where a network of computers can work on different parts of a task; however there are still limitations regarding the number of machines and data storage provided by a cloud network.

The limitations mentioned above encourage endeavors for more efficient use of resources in order to improve performance as much as possible, and that is where parallel processing becomes important. That is, parallel contribution of different processors (on a single computer), or different computers (on a cloud network) to a problem speeds the performance of a computational task up. In this procedure each of the parallel components provides a partial result, which needs to be integrated with that of other components in order to figure out the final result. The stage of integrating the results from different components of a computation task is called *reduction*.

The concept of parallel processing has been vastly used in network based applications, but has recently become an interest for shared memory applications (i.e. on single computers). A basic search about reduction on internet provides us with numerous links to articles that are mostly focused on network applications (e.g. chih Yang et al. [June 2007], Abouzeid et al. [August 2009]). For example, search based algorithms used by Google and Hadoop exploit reductions extensively in order to integrate the search results that are returned from several nodes (i.e. computers) in their network into one final result (Lammel [2008], HadoopWebsite [2014]).

However, as the potential for using reduction on shared memory applications is increasingly growing, the demand for rich assistant libraries has not been fulfilled reciprocally by different programming languages. OpenMP is one library which has been providing this feature for C/C++ and Fortran programming languages. Nevertheless, its support for reduction is limited about which we have elaborately explained in later sections of this paper. Moreover, lack of such a library is even more obvious for programming languages such as Java, as Java support for reduction is very primitive (JavaWebsite [2014]).

We have integrated complex reduction approaches that are normally used by network applications with simple trivial reductions, and have provided an extensive reduction library that can be used for both shared memory and network cases. Moreover, we have considered object oriented principles in order to let programmers flexibly extend the reduction functionality.

In Section 2. we have discussed some of the common reduction approaches and their use-cases. In Section 3. we

have overviewed the reduction provided by OpenMP, and have further discussed its limitations. In Section 4. our design approach has been elaborately explained, and few use-case examples have been provided in Section 5. in order to clarify the advantages of our design. Finally, in Section 6. we have discussed the outcomes of this paper as the conclusion.

2 Common Reduction Paradigms

References

Azza Abouzeid, Kamil Bajda-Pawlikowski, Daniel Abadi, Avi Silberschatz, and Alexander Rasin. Hadoop db: An architectural hybrid of mapreduce and dbms technologies for analythic workloads, yale university and brown university, us. *ACM VLDB '09*, pages 922–933, August 2009.

Hung chih Yang, Ali Dasdan, Ruey-Lung Hsiao, and Stott Parke. Map-reduce merge: Simplified relational data processing on large clusters. *ACM*, pages 1029–1040, June 2007.

HadoopWebsite. Hadoop reduction api. Available at Hadoop website: `http://hadoop.apache.org/docs/r2.4.1/api/org/apache/hadoop/mapred/Reducer.html`, 2014.

JavaWebsite. The java tutorials – aggregation operations - reduction. Available at: `https://docs.oracle.com/javase/tutorial/collections/streams/reduction.html`, 2014.

Ralf Lammel. Google's mapreduce programming model – revisited. *ELSEVIER Sience of Computer Programming*, 70:1–30, 2008.