MapReduce Principle for Spatial Data

F.-B. Mocnik

Vienna University of Technology, Gußhausstraße 27-29, 1040 Vienna, Austria
mocnik@geoinfo.tuwien.ac.at

1. Introduction

The amount of data accessible and used for spatial reasoning is rapidly increasing, in particular because efforts have been undertaken to make relations between data sets explicit, e. g. by the approach of linked data (Bizer, Heath and Berners-Lee 2009). It becomes increasingly hard to process data in acceptable time and complex reasoning is in many cases hardly possible. Data exhibiting these characteristics is called *big data* (Akerkar 2013, Lynch 2008, Snijders, Matzat and Reips 2012) and is relevant for GIScience (Adams, Brodaric, Corcho et al. 2012).

To address the processing of large amounts of data, parallelism becomes more important in order to use full processing power on multi-core and multi-processor computers as well as on computer clusters (Cannataro, Talia and Srimani 2002). MapReduce has been proven to be a successful approach that follows the paradigm of parallelism (Dean and Ghemawat 2004).

Libraries for applying MapReduce to spatial problems have already been implemented (Cary, Sun, Hristidis et al. 2009, Chen, Wang and Shang 2008, Eldawy and Mokbel 2013, Wang, Han, Tu et al. 2010). Research has been done on different aspects, like balancing the work between different cores (Chen, Chen and Zang 2010) and how to use spatial joins (Zhang, Han, Liu et al. 2009) and spatial indexes (Zhong, Han, Zhang et al. 2012) with MapReduce.

In this paper we discuss when and how MapReduce can and when it cannot be applied to spatial problems, and how to modify the problem in the latter case such that MapReduce can be applied to at least parts of the problem.

2. MapReduce Principle and Spatial Data

MapReduce is a principle to parallelize computations on large data sets: (i) the given computational problem P can be partitioned into several smaller ones P_i, (ii) the problems P_i can be computed in parallel (this step is called mapping) and finally, (iii) the results are combined using a reduction function (Dean and Ghemawat 2004). This principle works as long as two major requirements are met: first, the overall problem can be partitioned, and secondly, the overall result can inexpensively be computed based on the computations' results of the partial problems. Many problems do meet these requirements but others do not.

Spatial data refers to space. In the following, we only consider spatial data which can be associated to spatial regions. For example, to a region U we can associate the number of trees, objects depicted in a map or information about bus stops. The data $\mathcal{F}(U)$ assigned to a region U can be mapped to some data $\mathcal{G}(U)$ which is the result of the computational problem.[1]

Applying MapReduce to a *spatial* problem $P\colon \mathcal{F}(U) \mapsto \mathcal{G}(U)$ suggests itself to take advantage of its spatial structure: a partition of space $U = \bigcup_i U_i$ leads to data sets $\mathcal{F}(U_i)$ which can be used to formulate smaller problems $P_i\colon \mathcal{F}(U_i) \mapsto \mathcal{G}(U_i)$. (In fact, the mapping relation should be the same for all P_i. Thus, we omit the index and simply write $P'\colon \mathcal{F}(U_i) \mapsto \mathcal{G}(U_i)$.) In the reduction step, the resulting datasets are combined to the result of the main problem by computing $\mathcal{G}(U) = \bigoplus_i \mathcal{G}(U_i)$.[2]

[1]The notation $\mathcal{F}(U)$ is based on the mathematical concept of a presheaf.

[2]This notation of the MapReduce principle mentions the used indexes only implicitly: the keys in before the mapping correspond to the indices i, and after the mapping step there exists only one index, allowing to omit it.

3. Conditions for MapReduce on Spatial Problems

Computations can in general not be divided into smaller computations which can be processed in parallel (Bernstein 1966). For applying MapReduce to a spatial problem $P\colon \mathcal{F}(U) \mapsto \mathcal{G}(U)$, the following conditions are sufficient:

Condition (C1) There exists a partition $U = \bigcup_i U_i$, data sets $\mathcal{F}(U_i)$ associated to each U_i and a mapping $P'\colon \mathcal{F}(U_i) \mapsto \mathcal{G}(U_i)$.

Condition (C2) There exists an operation $\oplus$ such that $\mathcal{G}(U) = \bigoplus_i \mathcal{G}(U_i)$ and $\oplus$ is inexpensive to compute.

Even if the conditions are not met, one may be able to conclude the solution $P(\mathcal{F}(U))$ by only using the $\mathcal{G}(U_i)$ and some additional information like the U_i and/or additional spatial data. Thus, the conditions are not necessary for applying MapReduce. On the other side, if both conditions are met, we are able to apply MapReduce because condition (C1) implies the partition (induced by its spatial structure) of step (i) as well as the existence of the problems P_i of step (ii). Condition (C2) ensures that the solutions of the problems P_i can be combined to the solution of the original problem P because the following holds:

$$P(\mathcal{F}(U)) = \mathcal{G}(U) \overset{\text{(C2)}}{=} \bigoplus_i \mathcal{G}(U_i) \overset{\text{(C1)}}{=} \bigoplus_i P'(\mathcal{F}(U_i))$$

This proves the conditions (C1) and (C2) to be sufficient for applying MapReduce.

Consider the following examples:

Example. (1) *Number of objects of a certain kind.* $\mathcal{F}(U)$ is a collection of objects that are located in U. Consider the problem $P\colon \mathcal{F}(U) \mapsto \mathcal{G}(U)$ of counting the objects of a certain kind κ, e. g. trees or street crossings, i. e. $\mathcal{G}(U)$ is a number.

For applying MapReduce, we choose an arbitrary partition $U = \bigcup_i U_i$ and define $\mathcal{F}(U_i) \subset \mathcal{F}(U)$ to be the subsets of all objects located in U_i. The problems $P_i\colon \mathcal{F}(U_i) \mapsto \mathcal{G}(U_i)$ map a collection of objects to the number of objects of kind κ. As every object in $\mathcal{F}(U)$ is contained in one (and only one) $\mathcal{F}(U_i)$, we can define $\oplus$ as the addition:

$$\mathcal{G}(U) = \sum_i \mathcal{G}(U_i) =: \bigoplus_i \mathcal{G}(U_i).$$

(2) *Statistical properties.* Whenever a statistical property is a quantity q set into proportion to the area as statistical population (in this case the area of U), one may compute q using MapReduce and then divide the result after the reduction step (iii) by the area. Observe that the division is an additional step to the MapReduce principle.

4. Reformulation of Problems for MapReduce

If a problem does not meet condition (C2), one may modify it such that the property is met. In the following, we will discuss an example and its exemplary modification to finally formulate a rule which can characterize if a modification is optimal.

Example. (3) Let $\mathcal{F}(U)$ be time table information about when and which bus line does stop at which bus stop $s \in U$, and $\mathcal{F}(U_i) \subset \mathcal{F}(U)$ the corresponding subsets with all bus stops located in U_i. Assume that our computational problem $P\colon \mathcal{F}(U) \to \mathcal{G}(U)$ is to compute how many different bus lines meet pairwise, i. e. the number of (unordered) pairs (b_1, b_2) of bus lines where b_1 and b_2 meet at least at one stop.

The problem P cannot directly be solved by applying MapReduce because for combining the results of smaller problems in the reduction step (iii), the number of pairs for each U_i is not sufficient for not counting a pair several times if it is element of $\mathcal{F}(U_i)$ and $\mathcal{F}(U_j)$ with $i \neq j$, i. e. if two bus lines meet more than once.

To use MapReduce in spite of this we define (for an arbitrary partition $U = \bigcup_i U_i$) smaller problems $Q' \colon \mathcal{F}(U_i) \mapsto \mathcal{H}(U_i)$ which compute the pairs for U_i. Thus, we are able to compute all pairs $\mathcal{H}(U) = \bigcup_i \mathcal{H}(U_i)$ as the union of all sets of pairs. (Every pair occurs at most once in the union.) After this, we apply a function $r \colon \mathcal{H}(U) \mapsto \mathcal{G}(U)$ which counts the number of pairs. This function r is not part of MapReduce but is needed to compute the problem P.

$$\begin{tikzcd}
\mathcal{F}(U) \arrow[rr, "P"] \arrow[dr, mapsto, "Q"'] & & \mathcal{G}(U) \\
& \mathcal{H}(U) \arrow[ur, mapsto, "r"'] &
\end{tikzcd}$$

This principle can also be used in general: a function is executed after the MapReduce step, and/or one before. Commonly, there is more than just one possibility to choose these functions. To determine which choice is optimal for computational purposes, a detailed analysis of the algorithm regarding its performance on a specific computer system would be necessary. However, we can formulate a formal property that ensures that as much as possible of the problem can be computed in parallel.

Assume the situation of MapReduce with a problem Q' (which can be computed using MapReduce) and a function r executed afterwards as well as alternative choices $\widetilde{Q}$ and $\widetilde{r}$:

$$\begin{tikzcd}
\mathcal{F}(U) \arrow[rr, "P"] \arrow[dr, mapsto, "Q"] \arrow[ddr, mapsto, "\widetilde{Q}"'] & & \mathcal{G}(U) \\
& \mathcal{H}(U) \arrow[ur, mapsto, "r"'] & \\
& \widetilde{\mathcal{H}}(U) \arrow[u, mapsto, dashed, "\exists!"] \arrow[uur, mapsto, "\widetilde{r}"'] &
\end{tikzcd}$$

If the choice of Q and r meets the following criterion, it is ensured that Q computes at least as much as $\widetilde{Q}$, and thus as much as possible is computed in parallel:

Optimality Criterion (OC) For every choice $\widetilde{Q}$ and $\widetilde{r}$ with $P = \widetilde{r} \circ \widetilde{Q}$ and $\widetilde{Q}$ computable using MapReduce, there exists one (and only one) map $i \colon \widetilde{\mathcal{H}}(U) \mapsto \mathcal{H}(U)$.

In general there may not exist any Q and r that meets this optimality criterion. If however the optimality criterion is met for a certain Q and r as well as for $\widetilde{Q}$ and $\widetilde{r}$, there exists one (and only one) map $i \colon \widetilde{\mathcal{H}}(U) \mapsto \mathcal{H}(U)$ and one (and only one) map $\widetilde{i} \colon \mathcal{H}(U) \mapsto \widetilde{\mathcal{H}}(U)$. Thus, i is an isomorphism with $\widetilde{i}$ as its inverse.

In the example given above, we may e. g. define $\widetilde{\mathcal{H}}(U)$ as a collection of unique identifiers which correspond to each possible tuple of bus lines. The functions i and $\widetilde{i}$ correspond to the translation between the pairs and the unique identifiers.

5. Conclusion

MapReduce cannot be applied to some spatial problems whilst it can be applied to others. We did formulate two sufficient formal conditions for being able to apply MapReduce. If these conditions are not met and MapReduce cannot be applied, one may be able to reformulate the problem such that MapReduce can be applied to a part of it. By the given optimality criterion, we are able to ensure that as much as possible of the computational problem is parallelized. This could be especially useful for formal verification of performance properties and as break condition of automatic parallelization algorithms.

References

Adams B, Brodaric B, Corcho O et al., eds., 2012, *Proceedings of the Workshop on GIScience in the Big Data Age. In conjunction with the 7th International Conference on Geographic Information Science (GIScience) 2012*.

Akerkar R, ed., 2013, *Big Data Computing*. Chapman and Hall/CRC, London.

Bernstein AJ, 1966, Analysis of Programs for Parallel Processing. *IEEE Transactions on Electronic Computers*, EC-15(5):757–763.

Bizer C, Heath T and Berners-Lee T, 2009, Linked Data - The Story So Far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22.

Cannataro M, Talia D and Srimani PK, 2002, Parallel data intensive computing in scientific and commercial applications. *Parallel Computing*, 28:673–704.

Cary A, Sun Z, Hristidis V et al., 2009, Experiences on Processing Spatial Data with MapReduce. In: *Proceedings of the 21st International Conference on Scientific and Statistical Database Management (SSDBM) 2009*, 302–319.

Chen Q, Wang L and Shang Z, 2008, MRGIS: A MapReduce-enabled High Performance Workflow System for GIS. In: *Proceedings of the 4th International Conference on eScience 2008*, 646–651.

Chen R, Chen H and Zang B, 2010, Tiled-MapReduce: Optimizing Resource Usages of Data-parallel Applications on Multicore with Tiling. In: *Proceedings of the 19th International Conference on Parallel Architectures and Compilation Techniques (PACT) 2010*, 523–534.

Dean J and Ghemawat S, 2004, MapReduce: Simplified Data Processing on Large Clusters. In: *Proceedings of the 6th Symposium on Operation Systems, Design and Implementation (OSDI) 2004*.

Eldawy A and Mokbel MF, 2013, A Demonstration of SpatialHadoop: An Efficient MapReduce Framework for Spatial Data. In: *Proceedings of the 39th International Conference on Very Large Data Bases (VLDB) 2013*.

Lynch CA, 2008, Big data: How do your data grow? *Nature*, 455(7209):28–29.

Snijders C, Matzat U and Reips UD, 2012, Big Data: Big Gaps of Knowledge in the Field of Internet Science. *International Journal of Internet Science*, 7(1):1–5.

Wang K, Han J, Tu B et al., 2010, Accdelerating Spatial Data Processing with MapReduce. In: *Proceedings of the 16th International Conference on Parallel and Distributed Systems (ICPADS) 2010*, 229–236.

Zhang S, Han J, Liu Z et al., 2009, SJMR: Parallelizing Spatial Join with MapReduce on Clusters. In: *Proceedings of the International Conference on Cluster Computing and Workshops (CLUSTER) 2009*.

Zhong Y, Han J, Zhang T et al., 2012, Towards parallel spatial query processing for big spatial data. In: *Proceedings of the 26th International Parallel and Distributed Processing Symposium Workshop and PhD Forum (IPDPSW) 2012*, 2085–2094.