Winter 12-8-2014

Approximate Analytics for The Cloud

Michel De Rougemont

APPROXIMATE ANALYTICS FOR THE CLOUD

Michel De Rougemont, University Paris II & Liafa-Cnrs, France, m.derougemont@gmail.com

ABSTRACT

We model the cloud as a network of servers holding large XML trees, where the communication costs between servers are high and the local computations costs are low. We propose a general method, *StatsReduce,* which combines some statistical information of the data on each server and construct a global statistics for the combination of the trees. We show how to use this global statistics to approximately answer Analytics queries on the global data, in the case of a composition of trees. The value of new services on the cloud is dependent on the efficient estimation of Analytics queries with methods such as *StatsReduce.*

Keywords: Cloud, approximation, analytics.

INTRODUCTION

We introduce a model of the cloud as a network of servers where each server holds a large XML tree t as in Figure 1. Servers can exchange information but the communication cost is large, whereas the computation cost on each server is small. There is a canonical way to compose these trees t_i with distinguished elements and obtain a large tree T, which we view as the *big data*. Figure 2 gives an example of such compositions.

In this model, we can do all the classical operations on each server, but we can only do limited operations between servers, i.e. exchange very limited data. We study OLAP (OnLine Analytical Processing) queries over the large T and we can prove that the servers must exchange large amount of data in order to answer such queries. Assume the tree t_1 stores sales records of complex products and an OLAP query such as *the analysis of the total sales per country,* whose answer is a distribution with support a set of country. The exact answer may be the distribution *China: 0.5, U.K. 0.3, U.S. 0.2*, represented as a piechart in Figure 4.

How could we answer Analytics queries on the large T, such that each server exchange only a finite amount of information? We show that it is feasible if we relax the exact answer for *approximate answers*. If we compute the distribution *China: 0.49, U.K. 0.31, U.S. 0.2,* the relative error with the exact distribution is 0.02, the L_1 distance between the two distributions. The question we study is the following: given some precision ε, can we ε-approximate OLAP queries on T such that each server exchanges a finite information with the other the servers ?

We propose a general method where each server computes some statistics of its data, and the servers only exchange these statistics. We use the term *StatsReduce* in [3], to contrast the method with the *MapReduce* method where big data are *Mapped* on servers and computations are *Reduced* along the networks.

We argue that Business models for the cloud need these techniques, as Analytics queries on the large T are very important and real communication costs are large. We use various models of approximate computations. Boolean queries are approximated as in Property Testing [1,2] for the Edit distance, Unary queries for Xpath queries are approximated as densities and OLAP queries are approximated as distributions.

In the first section, we describe our model of the cloud and the *StatsReduce* method. In the second section we study the approximation of OLAP queries and we conclude in third section.

MODELS OF THE CLOUD

In our model, a network of servers defines a *big data* T, as the composition of local data. In this setting, the local data is a tree t_i which we compose as larger trees in Figure 2 or as DAG (Directed Acyclic Graphs) in Figure 3. We use samples of the tree with some specific relationships.. A *sample* is a random subtree at depth *k* when the root is taken uniformly over the nodes of the tree, and where the depth is the distance over the vertical and horizontal axes. Given two samples s_1 and s_2 either there are on the same paths and one is the ancestor of the other (the vertical order $<_v$), or one is on a path which precedes the other (the horizontal order $<_h$).

We keep N samples, where N is independent of the size n but depends on the precision ε. Figure 1 shows a large XML tree with two distinguished leaves and 5 samples, in the order of the document. Notice that $s_1 <_h s_2$ and $s_2 <_v s_3$ and the union of the two orders is a total order. Distinguished leaves are some specific samples which allow the composition of trees. If t_1 is a tree with 2 distinguished elements, t_2 and t_3 two trees, then T= $t_1(t_2, t_3)$ is well defined. It is the tree t_1 where the first distinguished element is replaced by the root of t_2 and the second distinguished element is replaced by the root of t_3 .

This model is realistic if we stream a large XML tree. We can save N samples and their relationship online. We use a variation of the Vitter's reservoir sampling [4], keep a list of N selected nodes, and keep a stack for each path of the tree. When we start, the list contains the N-th first nodes. When we encounter the i-th new node (open tag) where i>N, we choose to keep it with probability i/N. If we keep it, we remove one of the nodes in the list with probability 1/N and replace it with the new node. We know the vertical and horizontal relationships of this new node with its predecessors, looking at the stack of the current path. The stack will allow to maintain the subtree at depth k from a node and to sample more nodes between this node and the root of the tree . Distinguished leaves are kept as samples, i.e. with their vertical and horizontal relationships.

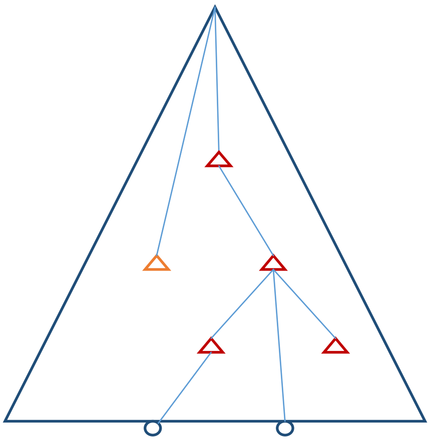

Figure 1: 5 samples and 2 Distinguished leaves

These samples with their relationships can be thought as a statistical representation of a large tree. The main result is that we can combine these statistics, as we build big trees T by composition of large trees, as T= $t_1 (t_2 , t_3)$ in Figure 2, and define a statistical representation of T.

The StatsReduce method [3].

Given a composition of trees, such as the one given in the Figure 2, how do we combine statistics? We need a robust representation such that small editions (insertions, deletions of nodes and edges) do not change the statistics. For each tree t_i the statistical representation is a graph G_i whose set of nodes is the set of samples, with two binary relations, the horizontal order $<_h$ and the vertical order $<_v$. A spanning tree for G with the relation between s_i and s_{i+1} allows to reconstruct G_i .

Two trees close for the Edit distance will have close densities for *p* sequences of subtrees of depth *k*, for *p*=1,2,…

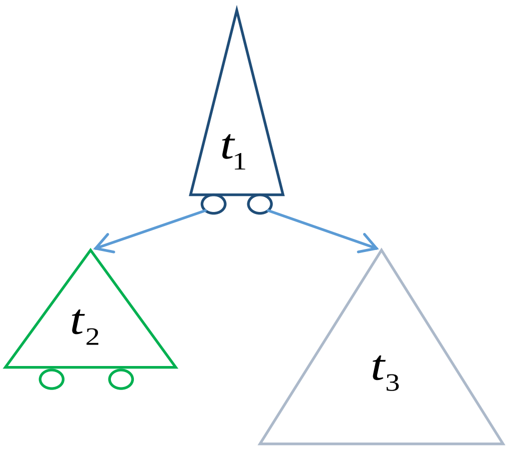

Figure 2: Tree of trees

Given a composition T= $t_1 (t_2 , t_3)$ as in Figure 2, assume t_i of size n_i has N_i samples and their relationships. Let $\mu=Min_i \{ N_i / n_i \}$, i.e. the minimum sampling rate in t_1 ,t_2 , t_3 . Assume t_1 has the minimum μ. We can then uniformy remove some of the samples in t_2 , t_3 in order to have the same rate in all the trees. If t_2 has 1000 samples and we only need 500, we randomly select them among the 1000s. The new graph G_T can then be built as the distinguished elements of G_1 are replaced by the roots of G_2 and G_3 and we extend the vertical and horizontal orders. We write $G_T = G_1 (G_2 , G_3)$. This construction guarantees that:

- The samples are uniformly distributed in the composition
- The relationships between two nodes of G_T is uniquely determined.

If we compose the trees as in Figure 3, we build a DAG (Directed Acyclic Graph), and we can similarly compose some statistics.

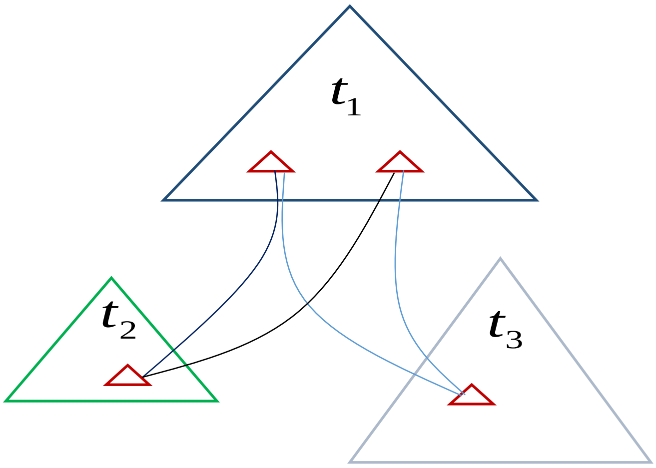

Figure 3: DAG of trees

2. APPROXIMATE ANALYTICS QUERIES

We first describe the classical OLAP queries on a tree T in this context, describe some examples and discuss the *StatsReduce* method which will produce approximate answers when servers exchange some finite information only.

Classical OLAP queries.
An OLAP query is specified by:

- A selection of the nodes: Xpath queries such as /a or /a//b are standard selection queries,
- Dimensions: Labels with discrete attribute values: Country with attributes values such as {China, U.S., U.K.},
- Measures and Operators: Labels with numerical values such as *<Sales value=120.5>* and an Operator (Sum).

The answer to an OLAP query is a distribution defined as follows: we first select the set S of nodes which satisfy the selection criterium. We then partition S according to the attribute values of the dimension. For each node of the partition we take the sum of the measures. The distribution is then obtained with support the set of attribute values and probability the relative sum of the measures.

A sample is a subtree s_i at depth k with several nodes with labels and attributes. We distinguish the labels with discrete (possible dimensions) from the labels with numerical attributes (possible measures).

Examples.
Consider t_1 as a sequence of sales records, with labels such as *Product*, *Type*, *Country*, *Sales.* The label *Type* has a discrete attribute *value*, in {hardware, software}, *Country* has a discrete attribute *name* in {China, U.S., U.K.}, and *Sales* has a numerical attribute *price* in [0,1000]. We can then ask for the Analysis per *Country* (dimension) of the total *Sales* (measure) of *Type hardware* (selection) products. A possible result could be *China: 0.5, U.K. 0.3, U.S. 0.2* also given in Figure 4.

If some of the products may contain other products, we may have the situation of Figure 2 where t_2 and t_3 represent similar records of nested products. In this case, we may ask for the analysis of *hardware* products per *Country*, which contain a subproduct made in *China*. It is a typical selection Xpath query of the type /Type[hardware]//Product/Country[China] and it is far more difficult to evaluate in this model, as the servers may need to exchange large data .

If the Sales records in t_1 also contain a label *Clients*, it is entirely feasible to have the label node *Type* pointing to a node in t_2 and the label node *Clients* pointing to a node in t_3 . We have then a Directed Acyclic Graph (DAG), as in Figure 3.

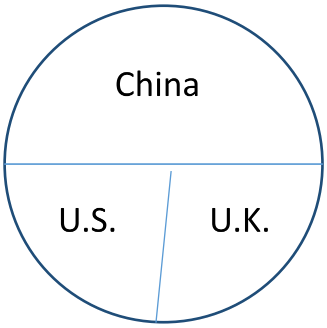

Figure 4: Answer to an OLAP query

Approximate Analytics with the StatsReduce method.
We assume that the graph G_T is built on the server which contains the root of the large tree T. Only some finite information (G_i) is exchanged between the servers because the number of samples N_i is independent of n_i, the size of the tree. Each sample s of G_T may contain numerical attributes and we can evaluate an OLAP query on the samples by doing:

- A selection on the nodes of G_T for the Xpath query
- A partition on the selected nodes of G_T on the dimensions
- An Aggregation of the measures on the samples of the partition.

We then obtain a distribution as an answer. How can we evaluate if it is a good approximation of the exact answer? The first necessary condition is to have a large set of nodes after the selection. If it is not the case, no guarantee can be given to the method. If we have a large set of nodes after selection, we can then show that each component of the distribution will be close to its mean, the real answer. A Chernoff-Hoeffding bound will guarantee that very few samples will suffice to make the error smaller than ε/D. With D distinct non zero components, we apply a Union bound and prove that the global error is less than the precision ε.

Examples(continue) .
For the approximate Analysis of $T= t_1 (t_2 , t_3)$ per *Country* (dimension) of the total *Sales* (measure) of *Type hardware* (selection) products, we construct G_T and apply the selection. If the relative density of the nodes is large, we partition the selected nodes and take the sum of the measure (*Sales prices*) on the samples. For the analysis of *hardware* products per *Country*, which contain a subproduct made in *China*, we proceed similarly.

In practice, suppose we fix some general sample rate, for example 10^{-4}. On a tree t_i of size 10^8, we have 10^4 samples. If we fix k = 4, the number of distinct subtrees among the 10^4 is an important parameter If there are all different, we are close to random data with no interest. In a real situation, we would have $3.\ 10^3$ distinct samples, hence $7.\ 10^3$ repetitions which would provide meaningful estimates of densities of *p* samples in some order. These density vectors have huge dimensions, but are sparse: most of the coefficients of these vectors are 0 or very small, so we can still reduce the support to its essential part, by ignoring the small coefficients. We may also concentrate on samples with specific labels at the root, reducing the support even more. The useful support size could be 500 for pairs of samples (necessary for the Xpath selection a//b). If we have a cluster with 100 nodes, the size of the *big data* T would be 10^{10} and the servers would exchange vectors of size $3.\ 10^3$. Within 100 steps we have the global statistics of T. The method would therefore give some estimate to many analytic queries.

CONCLUSION

The development of new services for the cloud requires a technology for Analytics queries over a combination of data hold on each server. We studied the case of trees with distinguished elements which compose to a large tree T, and showed how to use N_i samples on server i, to build a statistical representation G_i such that these representations compose. We built the graph G_T when $T= t_1 (t_2 ,t_3)$. Given an OLAP query on T, we construct from G_T an approximate answer, and the servers only exchange statistics, hence a small amount of information. This *StatsReduce* method can be applied to other contexts.

REFERENCES

[1] Fischer, E., Magniez, F., & de Rougemont., M. (2010) 'Approximate Satisfiability and Equivalence', *SIAM Journal of Computing*, Vol. 39, No. 6, pp.421–430.

[2] O. Goldreich, S. Goldwasser, & D. Ron. (1998) 'Property testing and its connection to learning and approximation', *Journal of the ACM*, Vol. 45, No. 4, pp.653–750.

[3] de Rougemont, M. (2014) 'StatsReduce in the cloud for Approximate Analytics', *The 2014 International Conference on Data Science and Advanced Analytics*.

[4] Jeffrey S. (1985) 'Vitter. Random sampling with a reservoir', *ACM Trans. Math. Softw.*, Vol. 11, No. 1, pp. 37–57.