StatsReduce in the cloud for Approximate Analytics

Michel de Rougemont
University of Paris II
LIAFA-CNRS
Email: m.derougemont@gmail.com

***Abstract*—We consider a cloud as a cluster of processors holding each a large XML tree. We present a statistical representation which can be built online on each processor and allows to approximate boolean, unary and Aggregation queries. The main result of the paper shows how these statistics can be efficiently *Reduced* to a master node of the cloud. We obtain an approximation of the global tree structure built from the elementary trees on each processor. In this *StatsReduce model*, processors only exchange statistical data with their neighbours. This technique leads to the approximation of Analytics queries on the global tree structure with a quantified confidence.**

I. Introduction

We introduce a *cluster model* where the cluster is a network of processors, each processor holds large amount of data (XML trees) and the processors only exchange some statistical data with their neighbours. The goal is to build a good approximate statistics for the global tree T, built from the tree cluster and the local trees t_i. We call this construction the *StatsReduce* phase.

We wish to approximate Analytics queries over T by reducing some specific statistical information over each node of a cluster to a global information on a master node. We assume that each node i of the cluster holds a large tree t_i, viewed as an XML tree with attribute values which can be discrete or numerical but bounded. The *global tree* T is the tree structure defined by the tree cluster where each node i is replaced by t_i. We will never store T on a single node. We want to answer queries on the global tree T, by just reducing statistical information on the cluster nodes. The main difference with the classical MapReduce setting is that there is no Map operation and the Reduce operation only applies to statistical data. We start from any state, where each node i stores a large tree t_i with successor nodes $j_1....j_k$ and explain how to reduce the statistics of the trees $t_i, t_{j_1}, ...t_{j_k}$ to the statistics of the composed tree, i.e. t_i followed by the trees $t_{j_1}, ...t_{j_k}$.

We use several types of approximation. The main notion is the approximation of a *boolean query*, as "is a tree t valid for a DTD?". We use the Property Testing [2] setting with the Edit distance on trees, and there is a theoretical interest in this framework as testers were only known for a weaker distance [1], [7]. We then consider *unary queries* which return sets of nodes, as an Xpath query "a//b" (the set of nodes labeled "a" with a descendant labeled "b"). In this case we approximate the density of such nodes within an additive ϵ. Finally we consider *OLAP queries* which specify dimensions (attributes with discrete values), a measure (attribute with a numerical value) and an aggregation operator (Sum, Average, Max,...) and return a distribution. We approximate these distributions with the L_1 distance. All these approximations are achieved with high probabilities, because they are obtained by some simple randomized algorithms. For the "a//b" nodes selected in the previous query, assume the "a" node has a successor node <*country name="U.K."*> and a successor node <*sales value="100"*>. The Analysis of the total Sales (measure) per Country (dimension) for the "a//b" nodes might be the distribution $\{U.K. : 0.2,\ China : 0.3,\ U.S. : 0.5\}$. The support is the set of attribute values $\{U.K.,\ China,\ U.S.\}$ and the measure the relative sums.

In this paper, we emphasize the approximation of analytic queries, i.e. OLAP queries over the large T and only give the intuition for why the statistical representation is also useful for boolean and unary queries. We introduce the main statistical vectors $\mathsf{ustat}_k^p(t)$ for a large tree t and two integer parameters k and p, which depend on the precision ϵ and the *depth* of the selection query ($p = 2$ in our example). It generalizes a k-gram, or the density $\mathsf{ustat}_k(t)$ of local subtrees at depth k to p sequences of such subtrees with their global relationships in the tree. This vector of very large dimension can be approximated with N samples where a sample is a sequence of p independent local subtrees at depth k and N only depends on k, p and the support of the distribution, i.e. the number of distinct local subtrees for the discrete attributes. It is important that the global relationships of the p subtrees can be computed when the tree is a large XML stream. The numerical attributes of these N samples will be stored separately from the density vectors $\mathsf{ustat}_k^p(t)$ together with the null values.

We give an efficient algorithm to approximate an OLAP query on T, if T is ϵ-close to some DTD and if the density of the selection query is high enough, i.e. greater than $c.\epsilon$, assuming all the sources are built as streams. The OLAP analysis on the samples will be $d.\epsilon$ close to the OLAP query on T, for some constants c, d.

We decompose the results in three parts:

- We first approximate an OLAP query on a large tree t close to a DTD, based on the samples and the statistics (Theorem 1).
- We show how the statistics vectors $\mathsf{ustat}_k^p(t)$ can be composed on a cluster, first for words and then for trees (Theorem 2).
- We combine the two first results and show how to approximate OLAP queries on T (Theorem 3).

In the second section, we review the basic notions of Property Testing, statistics on trees and their approximation when the tree is a large online stream, Analytic queries and the *Stats-reduction* model. In the third section we study the

approximation of OLAP queries on a tree t. In the fourth section we describe the *Stats-reduction* first on words and then on trees. In the fifth section, we combine both methods to approximate OLAP queries on the large global tree and discuss the practicality of the method.

II. Preliminaries

We define some statistical information which is sufficient to approximate any regular properties of trees. By keeping the numerical values of the samples, we can also approximate Analytic queries. The paper emphasizes the reduction of these statistics on a cluster, in order to lift the approximation to the *very large structure* T.

A. Property Testing

Property Testing [2] is a framework for approximate decision problems with a distance between structures such as words or trees. In this paper we use the *Edit distance* on labeled unranked ordered trees, as words are a special case. Basic operations are *edges insertions, edge deletions, labels modifications* at a unit cost. The absolute distance between two trees t, t' is the minimum number of operations which transform t into t'. The relative distance $0 \leq dist(t,t') \leq 2$ is the absolute distance divided by the largest size (number of nodes) of t, t'. Let a property P of trees be a subset of trees. The distance of t to P is $dist(t,P) = Min_{t' \in P}\ dist(t,t')$.

A *query* is a function which given a node v returns the local neighborhood of that node. The query complexity of an algorithm is the number of queries used. Given a parameter $0 \leq \epsilon \leq 1$, an *ϵ-tester* [2] for a property P is a randomized algorithm which makes queries and decides if a structure satisfies the property P or if it is ϵ-far from satisfying the property P. A property is *testable* if there exists *tester* such that for all ϵ it is an ϵ-tester and the query complexity is independent of the size of the structure and only depends on ϵ. In general, we select random nodes $v \in \{1, ...n\}$ taken with a uniform distribution and a query returns the neighborhood of v at distance $k = 1/\epsilon$.

We use a statistical embedding of trees which generalizes the classical k-gram, into a vector $\mathsf{ustat}_k^p(t)$ vector for two integer values k and p. It generalizes the simple testers given in [1] with $\mathsf{ustat}_k^1(t)$ for $k = \frac{1}{\epsilon}$ and a weaker distance, the *Edit distance with moves*, where an additional basic *move* operation is allowed. A move takes a subtree s of t, disconnects it from a node v and reconnects it to a node v', at the same cost than an Edit operation. It is similar to Editing a graph. Testers for properties of trees have also been studied in [5] for other distances.

This embedding allows to us to approximate boolean queries, density queries for Xpath, and analytic queries. Its main property is that it can be *Reduced* in a cluster.

B. Statistics

We consider *Statistics of a structure* as a distribution of finite support. Simple randomized algorithms take an XML stream and output a good approximation of these distributions. We first present the statistics on words and extend them to ordered unranked trees.

1) Statistics on words: A word w of length n is a structure $(V_n, \Sigma, Succ, <, L)$ where $V_n = \{1, 2, ...n\}$, the relations $Succ$, $<$ are the standard Successor and order relations on V_n and $L : V \to \Sigma$ is a labeling function. Let $\mathsf{ustat}_k(w)$ be the *uniform statistics of order* k, i.e. the k-gram of a word w. The vector $\mathsf{ustat}_k(w)$ of dimension $|\Sigma|^k$ gives the density of the subwords u_i of length k. Let $\#u$ be the number occurrences of a subword u and $u_1, u_2, ...u_{|\Sigma|^k}$ a lexicographic enumeration of the u_i's.

$$\mathsf{ustat}_k(w) = \frac{1}{n-k+1}. \begin{pmatrix} \#u_1 \\ \#u_2 \\ ... \\ \#u_{|\Sigma|^k} \end{pmatrix}$$

We write $\mathsf{ustat}_k(w)[u_i] = \frac{\#u_i}{n-k+1}$ and it is also the probability that a random subword of length k is u_i. We can also interpret $\mathsf{ustat}_k(w)$ as the distribution over all u's observed on a random position $1 \leq i \leq n-k+1$ if $u = w[i].w[i+1]...w[i+k-1]$. This statistics vector is however not enough to approximate regular properties of trees for the Edit distance.

Suppose we take two random positions $i_1 \leq i_2$ and define the $\mathsf{ustat}_k^2(w)$ vector as the density of (u_{i_1}, u_{i_2}), knowing that u_{i_1} *precedes* u_{i_2}. It has dimension $|\Sigma|^{2k}$ and there are $\binom{2}{n-k+1} + n - k + 1$ possibilities, $\binom{2}{n-k+1}$ cases when $i_1 < i_2$ and $n-k+1$ possibilities when $i_1 = i_2$. More generally we can take p random positions $i_1 \leq i_2 .... \leq i_p$ and define the (k,p) uniform statistics $\mathsf{ustat}_k^p(w)$. There are $\binom{p}{n-k+1}$ possibilities for p distinct positions, $(p-1).\binom{p-1}{n-k+1}$ for $(p-1)$ distinct positions, and so on. Globally there are M such tuples and $M = \binom{p}{n-k+1} + (p-1).\binom{p-1}{n-k+1} + \binom{2}{p-2}\binom{p-2}{n-k+1} + ...\binom{1}{n-k+1}$.

Definition 1: The (k,p) *statistics* $\mathsf{ustat}_k^p(w)$ vector of dimension $|\Sigma|^{p.k}$ is the density vector of the sequences $u_{i_1}, u_{i_2}...u_{i_p}$ for p random positions $i_1 \leq i_2, ... \leq i_p$.

$$\mathsf{ustat}_k^p(w) = \frac{1}{M}. \begin{pmatrix} \#(u_1, u_1, ...u_1) \\ \#(u_1, u_1, ...u_2) \\ ... \\ \#(u_{|\Sigma|^k}, u_{|\Sigma|^k}, ...u_{|\Sigma|^k}) \end{pmatrix}$$

We enumerate all p sequences of u_i's and count their number of occurrences.

Example. Consider binary words, and $k = 2$. There are 4 possible subwords of length 2, which we take in lexicographic order. For the binary word $w = 000111$, $\mathsf{ustat}_2^1(w) = \frac{1}{5}(2,1,0,2)$, i.e. $\#00 = 2$, and the first component is $\frac{2}{5}$. If $p = 2$, $\mathsf{ustat}_2^2(w) = \frac{1}{15}(3,2,.....)$ of dimension 16, as there are 3 occurrences of $(00,00)$ and 2 occurrences of $(00,01)$. Notice that there are 0 occurrences of $(11,00)$. There are 15 pairs, $\binom{p}{n-k+1} = \binom{2}{5} = 10$ pairs of distinct positions and 5 pairs of identical positions.

This classical lemma will be useful.

Lemma 1: Given $\mathsf{ustat}_k^p(w)$, we can obtain $\mathsf{ustat}_k^{p'}(w)$ for $1 \leq p' < p$.

Proof: We just apply linear interpolations. Assume $k = 1, p = 2$. Given $\mathsf{ustat}_k^2(w)$ we can obtain $\mathsf{ustat}_k^1(w)$ by:

$\mathsf{ustat}_k^1(w)[u_i] = \mathsf{ustat}_k(w)[u_i] = (2.\,\mathsf{ustat}_k^2(w)[u_i, u_i] + \sum_{u_j \neq u_i} \mathsf{ustat}_k^2(w)[u_i, u_j] + \sum_{u_j \neq u_i} \mathsf{ustat}_k^2(w)[u_j, u_i])/M$ where M is a normalizing factor, the sum over all u_i of the numerators. This method generalizes for arbitrary $p' < p$. ■

2) Statistics on trees: Consider labeled ordered unranked trees, as finite structures:

$$t_n = (V_n, \text{Firstsuccessor}, \text{Nextsibling}, <_v, <_h, L)$$

The domain is the set V_n of n nodes, $\text{Firstsuccessor}, \text{Nextsibling}$ are the standard relations of degree 1 defining the first successor of a node v and the next sibling. The relation $<_v$ is the classical Ancestor partial tree order and $<_h$ is the *horizontal order*. Let $i <_h j$ (i is before j) if $\neg(i <_v j)$ and i is before j in the XML document order. Notice that $<_v \cup <_h$ is a total order, called the order of the document.

We generalize the previous construction with ustat_k^p for trees. We take p random nodes, their subtrees at depth k and their horizontal or vertical relationships. The ustat_k^p vector gives the density of these p subtrees of depth k.

As for words $L : V_n \to \Sigma$ is a labeling *partial function* and Σ is a finite set of labels, the set of "tags and discrete attribute values". The expression *country name="U.K."* is an element of Σ and could be $L(v)$ for a node v. To handle numerical values, for example rational values less than $\beta = 100$ (the set Q_β), we add another labeling *partial function* $L_{num} : V_n \to Q_\beta$. The expression *sales value="81.5"* is represented as $L(v)$=*sales value=""* and $L_{num} = 81.5$. An additional difficulty is to handle XML trees with distinguished leaves $l_0, l_1, ...l_k$. These distinguished elements are necessary to define the composition of trees. An XML tree with numerical attribute values in Q_β and m distinguished leaves is a structure $t_n = (V_n, Q_\beta, \text{Firstsuccessor}, \text{Nextsibling}, <_v, <_h, L, L_{num}, l_0, l_1, ...l_m)$

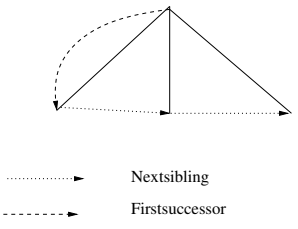

Fig. 1. Ordered unranked trees as a t_n structure.

A *subtree at depth* k from a node v includes all the nodes at distance less or equal to k for the relations Firstsuccessor, Nextsibling, as in Figure 2.

The number of distinct subtrees s_i of depth k is at most k^k and we can therefore enumerate all the γ labeled subtrees s_i of depth at most k and build the vector:

$$\mathsf{ustat}_k(t_n) = \frac{1}{n}.\begin{pmatrix} \#s_1 \\ \#s_2 \\ ... \\ \#s_\gamma \end{pmatrix}$$

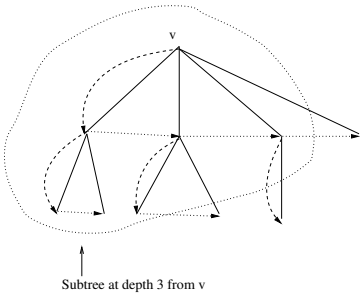

Fig. 2. A sample s_i at depth 3, from a random node v

where $\#s_i$ is the number of occurrences of the labeled tree s_i. We assume no numerical attributes. We will handle them later. The $\mathsf{ustat}_k(t_n)$ vector gives also the probability to observe s_i when a random node v is selected with the uniform distribution and the query selects the subtree of depth at most k rooted in v. If we select leaves, the depth of the subtree is 0.

Consider two distinct random subtrees s_i and s_j of depth k rooted in v_i and v_j and let $w = lca(u, v)$, the least common ancestor of the nodes u, v. There are two possible cases:

- s_i and s_j follow each other, if $lca(v_i, v_j) = v_i$ or v_j. We can decide this property by asking $i <_v j$.
- s_i and s_j are *horizontal* to each other, if $lca(v_i, v_j) \neq v_i$ and $lca(v_i, v_j) \neq v_j$. We say that s_i precedes s_j if $v_i <_h v_j$.

Let us write (s_i, s_j, v) if $v_i <_v v_j$ and (s_i, s_j, h) if $v_i <_h v_j$. The $\mathsf{ustat}_k^2(t_n)$ encodes the density of pairs of subtrees of depth k with their vertical or horizontal relationships.

$$\mathsf{ustat}_k^2(t_n) = \frac{1}{\binom{n}{2}}.\begin{pmatrix} \#(s_1, s_1, h) \\ \#(s_1, s_1, v) \\ \#(s_1, s_2, h) \\ \#(s_1, s_2, v) \\ ... \\ \#(s_\gamma, s_\gamma, v) \end{pmatrix}$$

If we select 3 subtrees we generalize to $\mathsf{ustat}_k^3(t_n)$ for $\#(s_1, s_2, s_3, h.v)$ if $\#(s_1 <_h s_2 <_v s_3)$ and all the other possible combinations. In general we define $\mathsf{ustat}_k^p(t_n)$ which gives densities for $\#(s_1, s_2, ...s_p, \sigma)$ where σ is a sequence of h and v of length $p - 1$.

C. Approximation of the ustat_k^p for words and trees

We always compare these vectors with the L_1 distance. Let $\widehat{\mathsf{ustat}}_k^p(t)$ be defined as ustat_k^p from $N = m.p = O(\frac{c'}{\epsilon^2})$ random samples which include the distinguished leaves, as follows. We take p uniform samples among the N, with their order and remove them from the list. From N samples we have N/p generalized samples with their order, on which we construct for $\widehat{\mathsf{ustat}}_k^p(t)$ as the vector ustat_k^p. We take the densities relative to the samples.

A sample is a group of p local trees $s_1, s_2...s_p$ at depth k, selected with their roots $v_1, ...v_p$ taken uniformly on V_n, such that $s_1 <_. s_2... <_. s_p$ where $<_.$ is either $<_h$ or $<_v$.

We build $\widehat{\mathsf{ustat}}_k^p(t)$ online as t is an XML stream. We need to show how we select N samples uniformly distributed, together with their relationships for $<_h$ and $<_v$. On a word w_n, this is the classical Vitter's reservoir sampling [8]. It takes a new letter $w[n]$ and maintains N sampled positions which we store in a sequence P.

Algorithm 1. (Uniform sampler) $(N, w[n])$:

1. For $n = 1, ...N$, store positions as a sequence $P = (1, ...N)$.
2. For $n > N$, flip a random bit $X \in_r \{0, 1\}$ such that $X = 1$ with probability $\frac{N}{n}$.
2.1. If $X = 1$, choose $Y \in_r \{1, 2, ...N\}$ such that $Y = i$ with probability $\frac{1}{N}$. Replace the i-th element of P by n.

An easy observation (by induction on n) is that the positions in P are uniformly distributed over $\{1, 2, ...n\}$ for all n. It generalizes to an XML tree where we read a new tag $t[n]$ and have two different actions for an open or a closed tag. We let b be the branching factor, i.e. the number of distinguished leaves in a general tree. The distinguished leaves behave as fixed samples on the leaves. We will maintain P as for the words, but also an ordered version Q with the specific orders. If $P = (17, 5, 32)$ than Q might be $(5 <_h 17 <_v 32)$.

Algorithm 2. (XML Uniform sampler)$(N, t[n])$:

We maintain a stack with all the positions and tags along the current path, a sequence P of positions of size N and a sequence B of size b of leaves positions.
1. For the $n = 1, ...N$ first opening tags, store positions in the sequence $P = (1, ...N)$.
2. For $n > N$, if $t[n]$ is opening, push $t[n]$ on the stack,
2.1. Decide if it is a distinguished leaf,
2.2. Flip a random bit $X \in_r \{0, 1\}$ such that $X = 1$ with probability $\frac{N}{n}$.
2.3. If $X = 1$, choose $Y \in_r \{1, 2, ...N\}$ such that $Y = i$ with probability $\frac{1}{N}$. Replace the i-th element of P by n. Update Q as we know which elements of P are on the current path.
3. For $n > N$, if $t[n]$ is closing, update the local subtrees of the samples and the stack.

Example. Consider an XML stream where we fix $N = 6$ samples and 2 distinguished leaves in advance. On the Figure 3, we observe the following (partial) relationships between the samples and distinguished leaves:

$$s_1 \leq_h s_2 \leq_h s_3$$

$$s_3 \leq_v s_4 \leq_v L_0$$

Consider the sequence of operations as we pass and mark s_4, L_0, s_5 in the stream. When we decide to select s_4, we know that $s_3 \leq_v s_4$ as we maintain the stack of the current path. We also know that $s_1 \leq_h s_4$ and $s_2 \leq_h s_4$ by observing Q at that point. When we decide that L_0 is a distinguished leaf, we know similarly that $s_3 \leq_v L_0$, $s_4 \leq_v L_0$. When we leave s_4 (closing tag) we have a complete description of the local subtree s_4 at depth k. When we enter s_5, we know $s_3 \leq_v s_5$ by observing the stack and $s_4 \leq_h s_5$ by observing Q.

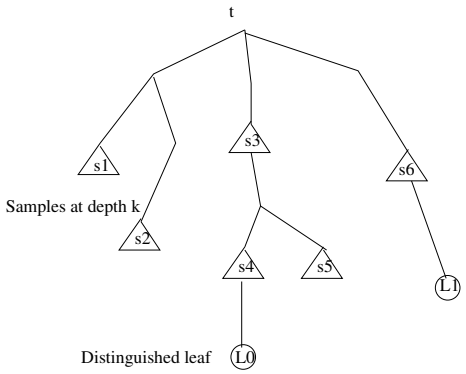

Fig. 3. XML stream: 6 samples, 2 distinguished leaves.

Lemma 2: For large trees t, the density $\widehat{\mathsf{ustat}}_k^p(t)$ ϵ-approximates $\mathsf{ustat}_k^p(t)$ with high probability.

Proof: For each component $[s_1, ...s_p]$, observe that the expectation $E(\widehat{\mathsf{ustat}}_k^p(t)[s_1, ...s_p, \sigma]) = \mathsf{ustat}_k^p(t)[s_1, ...s_p, \sigma]$. As we use N independent trials and bounded variables, we can use the Chernoff-Hoeffding bound [3] applied to such random variables Q:

$$Pr[\mid E(Q) - Q \mid \leq \epsilon/c] \geq 1 - 2.e^{-2.\epsilon^2/c^2.N}$$

Let $Q = \widehat{\mathsf{ustat}}_k^p(t)[s_1, ...s_p, \sigma]$ We take a union bound for all the c non-zero components $[s_1, ...s_p]$ components and obtain:

$$Pr[\mid \mathsf{ustat}_k^p(t) - \widehat{\mathsf{ustat}}_k^p(t) \mid \leq \epsilon] \geq 1 - 2.e^{-2.\epsilon^2/c^2.N}$$

Choosing N as indicated where c' is proportional to c, we obtain the result. ■

D. Approximate Analytics Queries

In the classical relational setting, an OLAP query is specified by a filter, some dimensions, a measure and an aggregation operator. We view the answer to an OLAP query as a distribution [6] which we approximate with the L_1 distance. As an example the analysis of sales per country in 2000, may yield:

$U.K.$	200
$China$	300
$U.S.$	500

We view the result as the distribution $\{U.K. : 0.2,\ China : 0.3,\ U.S. : 0.5\}$. A piechart is a standard representation of a 1-dimension OLAP query but generalizes to multidimensional piecharts for more dimensions.

There are several approaches such as [4] to generalize the OLAP analysis to trees. In our setting (XML trees), we assume the tree close to a DTD. It guarantees that most local subtrees have a structure with dimensions nodes and a measure node. A filter is defined by an Xpath query, the dimension is an attribute name, the measure is some attribute value. For simplicity, the aggregation operator will be the sum. We first show how to approximate an OLAP query on a tree t_n, as in [6] for relational OLAP schemas.

E. Tree clusters and Stats-Reductions

We now formally define the global tree T given a tree cluster where each node j with k successors holds a tree t_j with k distinguished leaves.

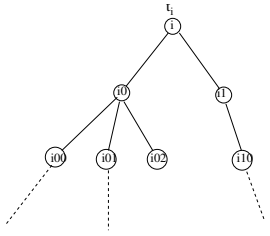

Fig. 4. Tree cluster of trees.

The global tree T is then defined as the composition of trees along the distinguished leaves of each t_i.

Example. For the tree cluster of Figure 4, the node i has two successors, $i0$ and $i1$. Hence t_i has two distinguished leaves L_0 and L_1. We can then replace L_0 by the root of t_{i0} and L_1 by the root of t_{i1} and write the new tree as $t_i(t_{i0}, t_{i1})$. If we keep this top-down construction we build the global tree T.

The *Stats-Reduction* method is a distributed algorithm where each node j of the tree cluster only exchanges statistical information with his neighbours in the cluster. In the example of Figure 4, node $i0$ can only exchange statistical information with nodes $i, i00, i01, i02$.

By opposition to the MapReduce method, we start with the large data on each server, so there is no Map operation. We don't reduce pairs of key-values, the Reduce operation, but *Reduce statistics* of the local data to a master node, the root of the cluster.

III. Approximation of an OLAP query on a tree t

We assume that all numerical values are bounded by a fixed value β. Without this hypothesis no approximation is possible as one very large hidden value could invalidate any approximation based on statistics.

We first need to extend the $\mathsf{ustat}_k^p(t_n)$ vector when the tree has numerical attributes. Assume some node label with a numerical such as *sales value="81.5"*. We list all the possible numerical values for a subtree s at depth k with numerical values, and similarly for the samples. A classical observation is that if we have a large enough number of samples, the OLAP query on the samples, with a limited number of numerical values will approximate the exact OLAP query.

Let num be a function which takes a subtree at depth k in t_n and a numerical attribute and list all the values, using "-" as a null value. For example

$$\mathsf{num}(s_i, value) = \{81.5, 73, -, 50.....\}$$

Similarly $\widehat{\mathsf{num}}$ lists the values of the samples

$$\widehat{\mathsf{num}}(s_i, value) = \{81.5, -, -, 6, 25\}$$

In this case we have 5 samples, 2 null values. We write $\sum \widehat{\mathsf{num}}(s_i, value)$ for the sum of the numerial value (112.5 in the example). We never store the function num but only store $\widehat{\mathsf{num}}$ The Sum aggregation is a linear function and all linear functions can be estimated from a limited number of samples.

Example. We analyze the total sales of the "a" nodes per country, assuming many "a" nodes have a *country.x* successor node for $x \in \{U.K., China, U.S.\}$ and a *sales.value* successor node. Suppose the exact answer is the distribution $\{U.K. : 0.2, China : 0.3, U.S. : 0.5\}$.

We look at the statistics of the s_i of the form $a(country.x, sales.value)$ for all values of x. We produce a distribution $\{U.K. : \delta_{U.K.}, China : \delta_{China}, U.S. : \delta_{U.S.}\}$. With $\widehat{\mathsf{ustat}_2}(t_n)$, we get three density values $\alpha_x = \widehat{\mathsf{ustat}_2}(t_n)[a(country.x, Sales)$ for the three values of x. Let $S_x = \sum \mathsf{num}(a(country.x, Sales), Value)$, i.e. the Sum Aggregation of the numerical values of the samples. Then

$$\delta_x = \frac{\alpha_x.S_x}{\sum_x \alpha_x.S_x}$$

and a typical result would be $\{U.K. : 0.18, China : 0.31, U.S. : 0.51\}$, within 4% of the exact solution.

An OLAP query on trees specifies a set of nodes (the base nodes) by a selection. The base nodes have a successor labeled by a dimension (country.name) and a successor node labeled by a measure (sales.value). The answer to the OLAP query gives the sum of the measure (sales) for each discrete value of the dimension.

This approach is consistent with the OLAP world such as [4] but supposes that the tree has some predefined structure captured by a DTD, and that the base nodes have been selected by a selection query, for example an Xpath query. It turns out that both these queries can be approximated. An OLAP analysis supposes a tree close to a DTD and a high density of the base nodes, defined by the selection.

Let the random variables $\alpha_x = \widehat{\mathsf{ustat}_2}(t_n)$[base(dimension.x,measure)] for the discrete values of x and $S_x = \sum \mathsf{num}$(base(dimension.x,measure), Value). The estimation of the OLAP query is given as follows:

Algorithm 3 (Estimation of an OLAP query). On input *(base, dimensions, measure, aggregation)*, do:

1. For each discrete value x of the dimension:
1.1 Compute $\alpha_x = \widehat{\mathsf{ustat}_2}(t_n)$ [base(dimension.x,measure)]
1.2 Compute $S_x = \sum \widehat{\mathsf{num}}$(base(dimension.x,measure), Value)
1.3 Compute $\delta_x = \frac{\alpha_x.S_x}{\sum_x \alpha_x.S_x}$

This procedure generalizes for neighborhoods at depth k, and for Xpath selections.

Theorem. 1: Given an OLAP query such that the density of the base nodes is large enough, the distribution defined by Algorithm 3 is an ϵ-approximation of the OLAP query with high probabilities.

Proof: Consider the expected value $E(\delta_x) = E(\frac{\alpha_x.S_x}{\sum_x \alpha_x.S_x})$. We can approximate the exact density, proportional to $\sum \mathsf{num}$(base(dimension.x,measure), Value) by the expected weight of the subtrees labeled

base(dimension.x,measure) times the average value of the $\widehat{\mathsf{num}}$ function, which is $\alpha_x.S_x$ normalized, i.e. $E(\delta_x)$. We can then apply the Hoeffding bound as in lemma 2 because all the numerical variables are bounded by β. The required number of samples depends on ϵ and β. ■

IV. Stats-Reductions for words and trees clusters

This first example on words serves as a motivation for the basic concepts which will be generalized to trees.

A. Words cluster

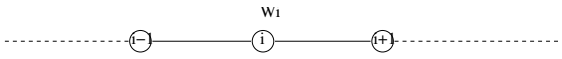

Fig. 5. Linear cluster of words.

Consider the ith node of the cluster holding the large word w_i of size n_i, with neighbours nodes $i-1$th and $i+1$th holding w_{i-1} of size n_{i-1} and w_{i+1} of size n_{i+1} We can approximate the statistics of $w_{i-1}.w_i.w_{i+1}$ from the individual statistics of nodes $i-1, i$ and $i+1$.

Lemma 3: $\mathsf{ustat}_k^p(w_{i-1}.w_i.w_{i+1})$ can be ϵ-approximated from $\widehat{\mathsf{ustat}}_k^p(w_i)$,$\widehat{\mathsf{ustat}}_k^p(w_{i-1})$ and $\widehat{\mathsf{ustat}}_k^p(w_{i+1})$.

Proof: Let us assume $p=2$, $w=w_{i-1}.w_i.w_{i+1}$ and $n=n_{i-1}+n_i+n_{i+1}$. We first compute $\widehat{\mathsf{ustat}}_k^1(w_j)$ for $j=i-1,i,i+1$ as in lemma 1. When we select two random subwords in $w_{i-1}.w_i.w_{i+1}$, the probability they fall in w_{i-1} is $\frac{n_{i-1}^2}{n^2}$, and the probability the first subword is in w_{i-1} and the second in w_i is $\frac{2.n_{i-1}.n_i}{n^2}$ and similarly for all the other cases.

If the two samples fall in w_{i-1}, the global statistics follow $\mathsf{ustat}_k^2(w_{i-1})[u_j,u_k]$, whereas if the first sample is in w_{i-1} and the second in w_i the global statistics is $\mathsf{ustat}_k^2(w)[u_j,u_k] = \mathsf{ustat}_k^1(w_{i-1})[u_j].\,\mathsf{ustat}_k^1(w_i)[u_k]$.

We can then interpolate the global statistics as:

$$\overline{\mathsf{ustat}}_k^2(w)[u_j,u_k] = \tfrac{n_{i-1}^2}{n^2}.\widehat{\mathsf{ustat}}_k^2(w_{i-1})[u_j,u_k]$$

$$+\tfrac{2.n_{i-1}.n_i}{n^2}.\,\mathsf{ustat}_k^1(w_{i-1})[u_j].\widehat{\mathsf{ustat}}_k^1(w_i)[u_k]+....$$

The sum has 3 components of the first type (cases $j-1,j,j+1$) and 3 components of the second type (cases $(j-1,j),(j-1,j+1),(j,j+1)$). It only ϵ-approximates ustat_k^2 as we miss the cases when the samples fall on the two borders $w_{i-1}.w_i$ and $w_i.w_{i+1}$. The impact is of order $1/n$. This interpolation method generalizes for an arbitrary p. ■

Similarly we can approximate the statistics of $w_{i-1}.w_i$ from the individual statistics of nodes $i-1$ and i.

Corollary 1: $\mathsf{ustat}_k^p(w_i.w_{i+1})$ can be ϵ-approximated from $\widehat{\mathsf{ustat}}_k^p(w_i)$ and $\widehat{\mathsf{ustat}}_k^p(w_{i+1})$.

Assume a linear cluster with $n=2^m$ processors holding $w_1,.....w_n$ and a master node (at position $n/2$) holding $w_{n/2}$. The merging of statistics can be done by dichotomy where the cluster nodes in position $2,5,8,....$ updates their neighbours statistics according to lemma 5. For the next stage they need to communicate at distance 3 in the cluster (node 5 need the statistics of nodes 2 and 8). If we assume that the main cost is the communication, we may as well simultaneously reduce the left and right branch following the corollary 1 until we reach nodes $n/2-1, n/2$ and $n/2+1$.

Lemma 4: The vector $\mathsf{ustat}_k^p(w_1,...w_{i-1}.w_i.w_{i+1}...w_n)$ can be ϵ-approximated from $\widehat{\mathsf{ustat}}_k^p(w_j)$ for $j=1,...n$ within $O(n/2)$ communication steps where only statistics vectors (constant size) are exchanged between nodes of the cluster.

B. Trees cluster

We generalize the previous construction with $ustat_k^p$ associated with trees. Consider the ith node of a tree cluster as in Figure 4, holding the large tree t_i of size n_i with two distinguished leaves (L_0,L_1), with first successor node $i0$th holding t_{i0} of size n_{i0} and and second successor node $i1$ holding t_{i1} of size n_{i1}. We show how to approximate the statistics of this subtree $t_i(t_{i0},t_{i1})$ from the individual statistics of nodes $i, i0$ and $i1$. Recall that the $\widehat{\mathsf{ustat}}_k^p$ is built from samples which include the distinguished elements. In particular we know all the samples s_j of $\widehat{\mathsf{ustat}}_k(t_i)$ such that $s_j <_v L_0$, $s_j <_h L_0$ and $L_0 <_h s_j$. Let $V_{L_0}=\{s_j : s_j <_v L_0\}$, $H_{L_0}^-=\{s_j : s_j <_h L_0\}$ and $H_{L_0}^+=\{s_j : L_0 <_h s_j\}$.

Lemma 5: $\mathsf{ustat}_k^p(t_i(t_{i0},t_{i1}))$ can be $2.\epsilon$-approximated from $\widehat{\mathsf{ustat}}_k^p(t_i)$,$\widehat{\mathsf{ustat}}_k^p(t_{i0})$ and $\widehat{\mathsf{ustat}}_k^p(t_{i1})$.

Proof: Assume $p=2$ and $n=n_{i0}+n_i+n_{i1}$. We first compute $\widehat{\mathsf{ustat}}_k(t_j)$ for $j=i0,i,i1$ as in lemma 1, generalized to trees. In particular we know all the samples s_i of $\widehat{\mathsf{ustat}}_k^p(t_i)$ such that $s_i <_v L_0$, $s_i <_h L_0$ and $L_0 <_v s_i$. When we select two random subtrees in $t_i(t_{i0},t_{i1})$, the probability they fall in t_{i0} is $\frac{n_{i0}^2}{N^2}$, and the probability the first subtree is in t_{i0} and the second in t_i is $\frac{2.n_{i0}.n_i}{N^2}$ and similarly for all the other cases.

If the two samples s_j,s_k fall in t_{i0}, the global statistics follow $\widehat{\mathsf{ustat}}_k^2(t_{i0})[s_j,s_k,.]$. If the first sample is in t_i and the second in t_{i0} it contributes to the vertical component of the global statistics if $s_j \in V_{L_0}$. Its contribution to $\mathsf{ustat}_k^2(t_i(t_{i0},t_{i1}))[s_j,s_k,v]$ is $\widehat{\mathsf{ustat}}_k(t_i)[s_j \in V_{L_0}].\widehat{\mathsf{ustat}}_k^1(t_{i0})[s_k]$. It contributes to the horizontal of the global statistics if $s_j \in H_{L_0}^-$. Its contribution to $\mathsf{ustat}_k^2(t_i(t_{i0},t_{i1}))[s_j,s_k,h]$ is $\widehat{\mathsf{ustat}}_k(t_i)[s_j \in H_{L_0}^-].\widehat{\mathsf{ustat}}_k^1(t_{i0})[s_k]$. If $s_j \in H_{L_0}^+$ its contribution to $\mathsf{ustat}_k^2(t_i(t_{i0},t_{i1}))[s_k,s_j,h]$ is $\widehat{\mathsf{ustat}}_k^1(t_i)[s_j \in H_{L_0}^+].\widehat{\mathsf{ustat}}_k^1(t_{i0})[s_k]$. The other cases are similar. We interpolate first the *vertical* component of the global statistics as:

$$\overline{\mathsf{ustat}}_k^2(t_i(t_{i0},t_{i1}))[s_j,s_k,v] = \tfrac{n_{i0}^2}{n^2}.\widehat{\mathsf{ustat}}_k^2(t_{i0})[s_j,s_k,v] + \tfrac{2.n_{i0}.n_i}{n^2}.\widehat{\mathsf{ustat}}_k^1(t_i)[s_j \in V_{L_0}].\widehat{\mathsf{ustat}}_k^1(t_{i0})[s_k]+....$$

The sum has 3 components of the first type (cases $i, i0, i1$) and 2 components of the second type (cases $(i,i0),(i,i1)$). We then interpolate the *horizontal* component of the global statistics similarly as:

$$\overline{\mathsf{ustat}}_k^2(t_i(t_{i0},t_{i1}))[s_j,s_k,h] = \tfrac{n_{i0}^2}{n^2}.\widehat{\mathsf{ustat}}_k^2(t_{i0})[s_j,s_k,h] + \tfrac{2.n_{i0}.ni1}{n^2}.\widehat{\mathsf{ustat}}_k^1(t_{i0})[s_j \in H_{L_0}^-].\widehat{\mathsf{ustat}}_k^1(t_{i1})[s_k]+....$$

The sum has 3 components of the first type (cases $i, i0, i1$) and 1 component of the second type (case $(i0, i1)$). Notice that $\overline{\mathsf{ustat}}_k^2$ only ϵ-approximates ustat_k^2 as we miss the cases when the samples fall on the borders $t_i.t_{i0}$ and $t_i.t_{i1}$. The impact is of order $1/n < \epsilon$. This interpolation generalizes for an arbitrary p. ∎

Consider a tree cluster with n_c nodes of depth d. Each cluster holds a tree t_i of size $n_i >> n_c$ and $n = \sum_i n_i$. Let $\overline{\mathsf{ustat}}_k^p(T)$ be the result of applying lemma 5 top down from the leaves to the root of the cluster.

Theorem. 2: If $n_c << n_i$ for each i, then $\overline{\mathsf{ustat}}_k^p(T)$ is an $2.\epsilon.d$ approximation of $\mathsf{ustat}_k^p(T)$.

Proof: We apply d times the construction of lemma 5 and obtain $\overline{\mathsf{ustat}}_k^p(T)$. At each stage we have an $2.\epsilon$-approximation for the corresponding $\mathsf{ustat}_k^p(T)$, so the global approximation factor is $2.\epsilon.d$. ∎

It is useful to have the $\widehat{\mathsf{ustat}}_k^p(t_i)$ built from a number of samples proportional to n_i the size of t_i. It is easier to show that the samples are uniformy distributed in the composition.

V. Analytic queries on the global tree T

We now combine the main results of the two previous sections. From Theorem 2, we can efficiently construct a good approximation $\overline{\mathsf{ustat}}_k^p(T)$ of $\mathsf{ustat}_k^p(T)$. Notice that although we can't construct T, we can have its statistics. From Theorem 1 we can approximate an OLAP query if we can maintain $\widehat{\mathsf{num}}(T)$. As in the composition step of lemma 5, define

$\overline{\mathsf{num}}(t_i(t_{i0}, t_{i1}))[s, value] = \widehat{\mathsf{num}}(t_i)[s, value] \cup_w \widehat{\mathsf{num}}(t_{i0})[s, value] \cup_w \widehat{\mathsf{num}}(t_{i1})[s, value]$, i.e. the weighted union of the sets of sampled numerical values for a fixed s and a numerical attribute. The weighted union takes the union for sizes proportional to the size of the trees. Suppose $n_{i0} = 2.n_{i1}$, and we have the same number of samples in both trees. In this case, we only keep a random half of the samples of t_{i1}. By induction on the depth of the cluster we obtain $\overline{\mathsf{num}}(T)$.

We need to show that $\overline{\mathsf{ustat}}_k^p(T)$ and $\overline{\mathsf{num}}(T)$ will give a good approximation to OLAP queries. Given an OLAP query on T, we first check using $\overline{\mathsf{ustat}}_k^p(T)$ that the density of the base nodes is high enough. We then replace in Algorithm 3, $\widehat{\mathsf{num}}(T)$ by $\overline{\mathsf{num}}(T)$.

Theorem. 3: Given an OLAP query on T such that the density of base nodes is large enough, the distribution defined by Algorithm 3 where $\overline{\mathsf{num}}(T)$ replaces $\widehat{\mathsf{num}}(T)$ is a good approximation with high probabilities.

Proof: The main argument is that $\widehat{\mathsf{num}}(T)$ and $\overline{\mathsf{num}}(T)$ have similar statistics. Samples taken uniformly in each tree t_i will be approximately uniform in T if for each i the number of samples in t_i is proportional to the size of t_i. We know that $\overline{\mathsf{ustat}}_k^p(T)$ is a good approximation. Hence for each x (value of the dimension), $\delta'(x)$ is close to the correct value $\delta(x)$. ∎

Although the method seems theoretical, we believe that practical applications may follow. 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 ustat_k, ustat_k^2 and ustat_k^3.

These vectors have huge dimensions, but they are sparse: $\widehat{\mathsf{ustat}_k}$ could have 3.10^3 non zero coefficients (its support), and ustat_k^2 much less than 9.10^6 non-zero coefficients because the statistics $\widehat{\mathsf{ustat}_k}$ would be non uniform. Most of the coefficients of these vectors would be 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 of $\widehat{\mathsf{ustat}_k}$ could be 500, the support size of $\widehat{\mathsf{ustat}_k^2}$ could be 10^3 and the support size of $\widehat{\mathsf{ustat}_k^3}$ could be 2.10^3. It is essentially all we need, with the num extension for the numerical values for each processor. If we have a cluster with 100 nodes, (the size of T would be 10^{10}), they exchange vectors of size 3.10^3.

Within 100 steps we have the global statistics. The method would therefore give some estimate to many analytic queries, and we would need to estimate the confidence for these values. Instead of starting with the approximation ϵ and confidence δ and infering the size of the representations (number of samples, parameters k, p), we would do the reverse. We keep a reasonable density of samples and infer the ϵ and δ.

VI. Conclusion

We presented the general *StatsReduce* framework for a cluster of processors, taking the trees as an example. This method can apply to other cloud situations, in particular to certain classes of graphs. We assumed that all the trees would be close to a common DTD. In practice, there may be several DTDs, one for each cluster which may store data associated with different languages (English, Chinese, Spanish...). In this case we need to combine data exchange techniques such as in [7], where the statistics are transformed before they are reduced.

References

[1] E. Fischer, F. Magniez, and M. de Rougemont. Approximate Satisfiability and Equivalence. *SIAM Journal of Computing*, 39(6):421–430, 2010.

[2] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM (JACM)*, 45(4):653–750, 1998.

[3] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(2):13–30, 1963.

[4] Mikael R. Jensen, Thomas H. Møller, and Torben Bach Pedersen. Specifying olap cubes on xml data. *J. Intell. Inf. Syst.*, 17(2-3):255–280, December 2001.

[5] Antoine Ndione, Aurélien Lemay, and Joachim Niehren. Approximate membership for regular languages modulo the edit distance. *Theor. Comput. Sci.*, 487:37–49, 2013.

[6] de Rougemont, M. and P. Thao Cao. Approximate answers to olap queries on streaming data warehouses. In *Proceedings of the fifteenth ACM international workshop on Data warehousing and OLAP (DOLAP)*, pages 121–128, 2012.

[7] de Rougemont, M. and A. Vieilleribière. Approximate data exchange. In *International Conference on Database Technology (ICDT)*, pages 44–58, 2007.

[8] Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, March 1985.