Journal of Southeast University (English Edition) Vol. 21, No. 3, pp. 293 – 298 Sept. 2005 ISSN 1003—7985

Data partitioning based on sampling for power load streams

Wang Yongli[1,2] Xu Hongbing[1] Dong Yisheng[1] Qian Jiangbo[1] Liu Xuejun[1]

([1]Department of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

([2]Department of Common Computer Teaching, Jiamusi University, Jiamusi 154007, China)

Abstract: A novel data streams partitioning method is proposed to resolve problems of range-aggregation continuous queries over parallel streams for power industry. The first step of this method is to parallel sample the data, which is implemented as an extended reservoir-sampling algorithm. A skip factor based on the change ratio of data-values is introduced to describe the distribution characteristics of data-values adaptively. The second step of this method is to partition the fluxes of data streams averagely, which is implemented with two alternative equal-depth histogram generating algorithms that fit the different cases: one for incremental maintenance based on heuristics and the other for periodical updates to generate an approximate partition vector. The experimental results on actual data prove that the method is efficient, practical and suitable for time-varying data streams processing.

Key words: data streams; continuous queries; parallel processing; sampling; data partitioning

The integration of parallel and approximate techniques is the inevitable developmental trend for data streams processing in the future. Continuous query (CQ) systems over data streams challenge the traditional data parallelism techniques because of the expiration of the data and the delay of the communication. They require adaptive, online repartitioning, and load balancing of lookup-based operators[1]. In many traditional parallel systems, the partition controller relies on histograms on selected columns to estimate partition range. Generating histograms and other statistical measures over large data sets are expensive propositions. Therefore, random sampling of data has been considered as a technique to efficiently construct approximate histograms.

In a parallel database domain, there are two fundamental approaches—exact splitting and approximate splitting to determine the data partition vector[2]. It is difficult to determine the quantiles exactly over infinite data streams, so we have to apply an approximate technique. The course of generating approximate quantiles can be regarded as the course of generating approximate equi-depth histograms. It is an expensive problem to look for buckets whose frequency variance is a minimum[3]. There are many existing sampling algorithms[4]. Reservoir sampling algorithm is adapted to sampling for the unknown sized files. Ref. [5] allowed skipping of a number of records when scanning original data sets. For on-the-fly data streams, the reservoir sampling algorithm is ideal sampling method. But this simple random sampling method does not adapt to the time-varying data streams. Each sample should not be selected with equal probability. Ref. [6] addressed this critical question of determining "how much sampling is enough" in the context of an equi-height histogram. However, its background work is statically large data sets. Moreover its equi-height histogram is applied to estimate the selectivity of range query. Our objective is to construct an efficient method of sampling and generating histograms adapted to data streams.

Received 2005-01-12.

Foundation items: The High Technology Research Plan of Jiangsu Province (No. BG2004034), the Foundation of Graduate Creative Program of Jiangsu Province (No. xm04-36).

Biographies: Wang Yongli (1974—), male, graduate; Xu Hongbing (corresponding author), male, professor, hbxu@seu.edu.cn.

1 Continuous Query Model and Definition

In the power industry, the value of load data streams generally obeys a certain approximate distribution. The partition vector produced from samples of data streams can portray the overall characteristic distribution of data streams. We can compute an approximate partition vector by sampling the data streams in a given time interval. Considering the expiration of data streams, the method based on range partition is more suitable for aggregate operation related content, which can effectively avoid the data skew for data repartition. Our strategy is to sample over data streams; apply an equi-depth histogram technique to generate an approximate partition vector over sample tuples; use the fixed size tuples in every bucket as partition granularity; and

apply the method of round robin range partition to redistribute tuples received locally to the other appropriator sites.

1.1 Parallel CQ processing model adapted to power load streams

According to the peculiarity of the power distribution network, our coordinator-worker model is parallel and distributed. Its architecture is composed of the coordinator and the worker with the query preprocess layer. Each site (including the worker and the coordinator) possesses its own local computational resources, and communicates via messages with each other. Compared with the existing SCADA system whose front-end is only responsible for acquiring data rather than processing complex tasks, this work-coordinator model distributes a lot of tasks to multiple intelligent front-end (worker) and intensively reduces the workload of the central server, so it can get better performance. For the details of the model, refer to Ref. [7]. The course of its data partitioning can be divided into three phases. The first and second phase of data partitioning are illustrated in Fig. 1 and Fig. 2. First, the coordinator generates a new partition vector using samples derived from each worker, and sends the partition vector to all workers. Secondly, the data streams streaming in a worker will be directly redistributed to the other workers according to the partition vector gained from the former phase. For example, at worker 1 in Fig. 2, original streams are redistributed to the rest of the workers. Other workers resemble worker 1 in the same way.

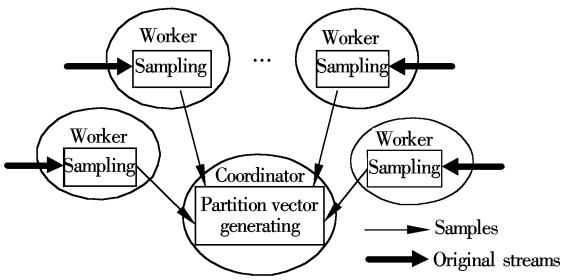

Fig. 1 First phase of data partitioning: sampling and generating partition vector

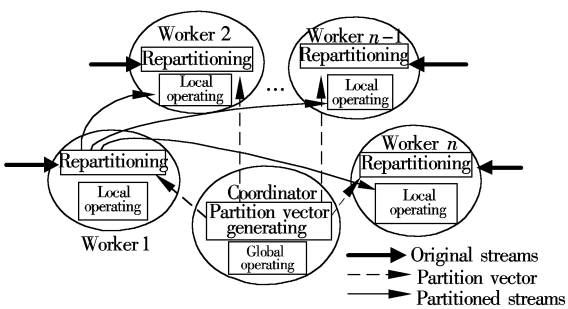

Fig. 2 Second phase of data partitioning: data distributing

Thirdly, after the query task and data streams have been distributed, we apply the repartition correction factor to implement load shedding dynamically. Since the third phase is not an emphasis of this paper, we omit it.

1.2 Relevant definition

Consider a stream S containing an integer-valued attribute X. The value set V of X is the set of values of X that are presented in S. For each $v \in V$, the frequency $f(v)$ is the number of tuples $t \in S$ with $t.X = v$. We assume that the elements of V have been sorted according to some sorting parameter, most commonly according to the numeric values of v_i, i.e., $V = \{v_i \mid 1 \leqslant i \leqslant N\}$ where $i < j$ iff $v_i < v_j$. Given this ordering, and using $f_i = f(v_i)$, the frequency set of X is the ordered set of frequencies $F = \{f_1, f_2, \ldots, f_N\}$.

A k-histogram[8] of data distribution X is constructed by partitioning the frequency set F of X into k ($\geqslant 1$) intervals called buckets. In each bucket, we approximate the frequencies and values in some succinct fashion. A k-histogram for V is said to be an equi-depth histogram if each bucket size, b_j, is exactly N/k. The approximate data distribution can be used in place of the actual distribution. Of course, the accuracy of any operation performed using the histogram depends on the accuracy of the approximation, which is determined by two factors, the partitioning technique employed for grouping values into buckets and the approximation technique employed within each bucket.

Definition 1 ε-approximate quantile

We say that a quantile summary is ε-approximate if it can be used to answer any quantile query to within a precision of εN (N denotes computing size). In other words, for any given rank r, an ε-approximate quantile summary returns a value whose rank r' is guaranteed to be within the interval $[r - \varepsilon N, r + \varepsilon N]$.

Definition 2 δ-deviant histogram

The maximum error metric for a k-histogram is defined as follows: $\Delta_{\max} = \max_{1 \leqslant j \leqslant k} |b_j - N/k|$, where b_j denotes the number of the tuple in the i-th histogram. A k-histogram with $\Delta_{\max} \leqslant \delta$ is said to be a δ-deviant histogram, and satisfies the property that for all j, $1 \leqslant j \leqslant k$, $|b_j - N/k| \leqslant \delta$.

We require that every bucket in the histogram has a size which has a small absolute difference with respect to the bucket sizes in an exact equi-depth k-histogram. Suppose that δ is equal to εN in our model.

Because there is a clear time-range semantic in the CQ for power load, when listening to new attributes involved in a new CQ, the new query plan will be carried

out over redistributed data. The users can receive the result after a tolerable period of delay. A typical continuous aggregation query for a power system can be constructed as follows:

SELECT COUNT($t_i.X$) FROM S WHERE VARIANT ($t_i.X - t_{i-1}.X$) > =500 WINDOW NOW - 30

This CQ language returns the number of tuples from stream S in the last 30 s time-window under the constraint—the difference of the X value in the adjacent two tuples is not less than 500 kW. For convenience, the notation used by this paper is described as follows: S is a stream; X is an integer-valued attribute in S; t is a tuple in S, $t \in S$; $t.X = v$; $f(v)$ is frequency (i. e. the number of v); V is the value set V of X; k is the number of sites (corresponds to the number of the bucket of the equi-depth histogram in this paper); N is the size of the time-window involved in CQ; n is the number of samples required; ε is the maximum approximate error; δ is the precision of constructing approximate an equi-depth histogram ($\delta = \varepsilon N$).

2 Sampling Method Adapted to Data Streams

We should consider the data distribution when we determine the number of tuples that require skipping over. This strategy can make up for the insufficient representative problem due to the case of different change of tuples in the different periods of time. We introduce the skip factor that records the character of data-value changing in data streams, and take a parallel samples for the same category data streams derived from the different sites based on a reservoir sampling algorithm.

The skip factor is the timed function based on historical data. The basic idea of the skip factor is, the more quick the data changing is, the more the number of tuples which are skipped. Let Δ_i denote the variety ratio of $t.X$ at i timestamp, $\Delta_i = \left| \frac{t_i.X - t_{i-1}.X}{t_{i-1}.X} \right|$. We define $\bar{\Delta}_i$ as the move average of Δ_i during $2m$ timestamps, $\bar{\Delta}_i = \sum_{i=-m}^{m-1} \Delta_i / 2m$, where m is the experiential value. The skip factor, named SF(i), is defined as:

$$\mathrm{SF}(i) = \begin{cases} \text{true} & \text{if } \bar{\Delta}_i > \eta \text{ (sampling)} \\ \text{false} & \text{otherwise (skipping)} \end{cases}$$

where i denotes the timestamp of tuples to be detected, and η is an experiential threshold.

Now we bound the number of random samples required to guarantee that the resulting histogram is δ-deviant. We give essentially optimal formulas based on theorem 4 in Ref. [6], describing the trade-off between the various parameters, notably k, δ, and n.

Let $\delta \leqslant N/k$, an equi-depth k-histogram for a random sample set R of size n from a value set V of size N gives a δ-deviant histogram for V with probability at least $1 - p$ ($p > 0$), provided that $n \geqslant \frac{4N^2 \ln(2N/p)}{k\delta^2}$. Or, equivalently, $n \geqslant \frac{4\ln(2N/p)}{k\varepsilon^2}$.

Algorithm 1 Improved reservoir sampling for data streams

Input: N, k and probability p.

Output: the array R contained n samples.

1) Compute the sampling number n, based on N, and k, p.

2) Initial sampling count j, $j \leftarrow 1$; pick a random sample, and store into $R[j]$.

3) Initial read pointer t for original stream, $t \leftarrow 1$.

4) Repeat

① Generate an independent random variable Y.

② Skip the next Y tuples in virtue of the skip factor.

③ If ($Y <= N$)/* select the next tuple to become a candidate; randomly replace some tuple in buffers */

a) $u \leftarrow$ (int)(n * (rand ())./* u obeys unique distribution and $u \in [0, n-1]$ */

b) Pick a random sample, and store into $R[u]$; $j \leftarrow j + 1$.

④ $t \leftarrow t + Y + 1$.

5) Until ($j > n$) or ($t > N$).

In this algorithm, step ② is related to the skip factor mentioned above. Due to limitation of space, we omit its details. The maximum time complexity of this algorithm is $O(n(1 + \log(N/n)))$.

3 Generating Equi-Depth Histogram Algorithm

The work that guarantees the number of tuples in every two adjacent quantiles is approximately corresponding to constructing an equi-depth histogram. We propose a heuristic generating equi-depth histogram algorithm and a shortcut effective generating equi-depth histogram algorithm.

Let BucketList denote the bucket array, BucketList[i] ($i \in [0, 1, \ldots, k-1]$) is a triple, BucketList[i] (left border, frequency, right border) consists of left border, the number of tuples in bucket and right border.

3.1 Heuristic method

Algorithm 2 Incremental maintaining heuristic

generating equi-depth histogram

Step 1 When a new tuple arrives, if it belongs to an existing bucket interval, then modify the number of this existing bucket. If it does not belong to an existing bucket interval and the total number of buckets has not reached k, then create a new bucket. The left and right borders of new buckets are $t.X-\delta$ and $t.X+\delta$, respectively, if overlapping happens at the borders of contiguous buckets, then shrink the left border of new bucket to the right border of the previous bucket and the right border of the new bucket to the left border of the next bucket. If the tuple does not belong to an existing bucket interval and the number of the buckets has already reached k, then insert it into the closest bucket.

Step 2 Clean up all the buckets by merging and splitting to meet equi-depth constraint. The principle of merging is to merge contiguous buckets containing a small quantity of tuples; the principle of splitting is to split the bucket containing a large quantity of tuples.

We can choose the median of the number of buckets to split. The difference between a large quantity and a small quantity is judged by the comparison between $|\text{BucketList}[i].\text{frequency}-N/k|$ and δ. If $|\text{BucketList}[i].\text{frequency}-N/k|>\delta$ then we say this bucket is "a large quantity"; otherwise, we say this bucket is "a small quantity". The time complexity of the algorithm is $O\left(\frac{1}{\varepsilon}Nk^2\right)$.

3.2 Shortcut method

If the maximum number of buckets and the total number of sampling data in every interval are already known, the method to generate an equi-depth histogram turns out to be relatively easy. We propose one shortcut effective approximate algorithm in order to assure that the difference between the number of tuples in the current buckets and the average number of tuples in all the buckets is minimum when producing each bucket border. The greedy tactics can achieve an approximately optimal histogram.

Algorithm 3 Periodical update shortcut generating equi-depth histogram

Step 1 Scan all the tuples in the sampling set, count the frequency of each attribute key, sort in ascending order, store these distinct keys and frequencies in distinct value[0, 1, …, $D-1$] array, where D denotes the number of distinct values, each unit of distinct value is a dual set (key, frequency).

Step 2 According to the precision and the maximum number of buckets, find the $k+1$ position among distinct values: 1, $1+N$, $1+2N$, …, $1+(k-1)N$, $1+kN=T$; the first is 1, the last is T, where $N=(T-1)/k$, assure that the number of key values between every two continuous positions is approximately equal, and the difference between the number of the bucket and N/k is the minimum.

For a given k, if the approximate precision is too high, it will result in failure for this strategy. We can set up the initial value of ε_0 according to experience, and then improve the precision and iteratively compute to find minimal error ε gradually.

Algorithm 3 offers adjustable precision. Its time complexity is $O(f(\varepsilon N)D\log(D))$, where $f(\varepsilon N)$ denotes the time used to reach minimal error ε from initial value ε_0.

4 Experiment Evaluation

In order to validate the proposed model, we use actual load data of a power system from one area in Nanjing to test practical performance of this model. We constructed the discrete event generator to simulate the AGC (automation generation control) system, which read a tuple at a fixed interval (10 ms) from the data sets and sent it to our model. There is an analogy between the way a power-distributed-network runs and the way this multiple data streams environment constructed by multiple generators runs. All data streams derived from the distributed worker (front-end) possess the same measured attribute. Suppose that the maximum error $\varepsilon=0.05$, $p=0.95$ and the number of buckets in the partition vector is 4, which is equal to the number of sites. We choose the aggregate operation mentioned above to test the algorithm.

The experiments were conducted on a 4-node shared-nothing cluster of 2.66 GHz Pentium machines in which every node had 256 MB main memory and a 80 GB hard disk. Connecting the machines was a 100 Mbit/s switched Ethernet network with a point-to-point bandwidth of 100 Mbit/s and an aggregate bandwidth of 800 Mbit/s in all-to-all communication. Each machine was booted with version 2.4.18 of the Linux kernel. For message passing between the Pentium nodes, we used the LAM implementation of the MPI communication standard. With the LAM implementation, we observed an average communication latency of 460 ms and an average transfer rate of about 10 Mbyte/s.

Experiment 1 validates the relationship between the number of samples and the execution time for gen-

erating an equi-depth histogram. Experiment 1 ran 10 trials on a single site, initializing the random number generator with a different seed each time.

Fig. 3 shows the running time used by our two kinds of generating equi-depth histogram algorithm and the existing algorithm (called MRL[9], which is the best previously known algorithm) over different numbers of input samples. The execution time in Fig. 3 is the mean of execution times. We can see that the execution time of the shortcut algorithm is shorter while its precision is lower. In contrast, the execution time of the heuristics algorithm is longer, while its precision is higher. The performance of the MRL algorithm is intervenient.

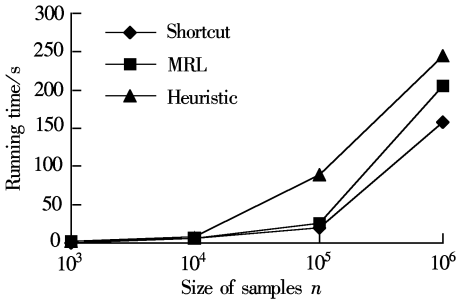

Fig. 3 Variation in running time vs. sampling size for shortcut, heuristic and MRL

The average time spent per stream tuple during the execution of this shortcut algorithm compared with the MRL algorithm is smaller, so our shortcut generating equi-depth histogram algorithm is suitable for online processing. We can adjust the error δ and the number of samples n to implement the trade-off between the results' precision and the computation velocity.

Experiment 2 studies the effect of the varying number of records in the stream on the required sampling size. We fixed the maximum error to 0.1, and varied the number of tuples between 3×10^4, 6×10^4, 18×10^4 and 36×10^4 (i.e. the length of the time-window in the query is 30, 60, 180 and 360 s, respectively). Fig. 4 shows that as the number of tuples in the stream increases, a proportionately smaller percentage of tuples need to be sampled to reduce the error below a given threshold. The behavior is consistent with the theoretical expression in section 3 which predicts that the required sampling size drops at a rate of $\log(N/n)$.

Experiment 3 validates the effect of the skip factor on the number of required samples. We observe the change of sampling size n derived from the same CQ in two kinds of period of time along with the load

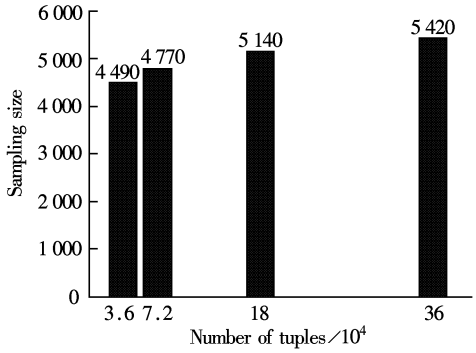

Fig. 4 Sampling size vs. the number of tuples (maximum error $\leqslant$0.05)

curve. Between 40 dot and 60 dot on the x-axis (corresponding to the wee hours 3:30 to 5:00), the change of load-value is gentle. We should pick the fewer samples. However, between 120 dot and 140 dot on the x-axis (corresponding the antemeridian hours 10:00 to 12:00), the verity of the load-value is quick. We should pick more samples. Suppose that the length of the time-window is fixed. Fig. 5 shows that the number of tuples that need to be sampled to achieve a given error threshold remains almost constant as the load data is streamed. This also conforms to the expected affection of the skip factor in section 3.

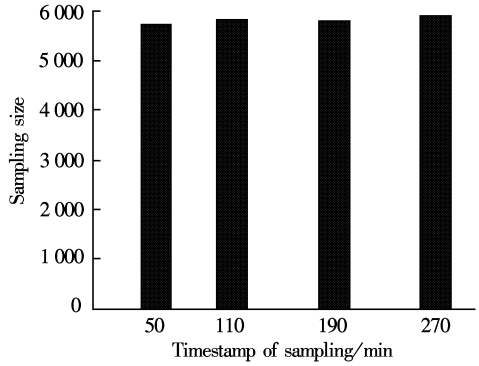

Fig. 5 Sampling size vs. sampling timestamp (length of time-window is 100 min)

5 Conclusion

We propose a novel data partition method using approximate techniques including sampling and an equi-depth histogram, which supports a variable repartition vector for dynamic load balancing. The experiments prove that the proposed model adapts to some application domains where the change of data-value distribution obeys a certain rule, such as industry control or traffic control etc. Our data partitioning method based on sampling can explicitly tailor the feature of data distribution, and can be more efficient and adaptive in a given application scenario. This model is a

feasible solution to online processing of time-varying data streams in short response time. Improvement of data streams algorithm efficiency and adaptive resource management over data streams are on our future research agenda.

References

[1] Shah M, Hellerstein J, Chandrasekaran S, et al. Flux: an adaptive partitioning operator for continuous query system [R]. Report No. UCB/CSD-2-1205, University of California Berkeley, 2002.

[2] Dewitt D J, Naughton J F, Schneider D A. Parallel sorting on a shared-nothing architecture using probabilistic splitting [A]. In: *Proc of the First International Conference on Parallel and Distributed Information Systems* [C]. New York: ACM Press, 1991. 280 - 291.

[3] Guha S, Koudas N, Shim K. Data streams and histograms [A]. In: *Proc of Symp on Theory of Computing* [C]. Heraklion, Crete, Greece: ACM Press, 2001. 471 - 475.

[4] Seshadri S, Jeffrey F. Sampling issues in parallel database systems [A]. In: 3*rd International Conference on Extending Database Technology* [C]. Vienna, Austria: Lecture Notes in Computer Science, 1992. 328 - 343.

[5] Vitter J S. Random sampling with a reservoir [J]. *ACM Transactions on Mathematical Software*, 1985, **11**(1): 37 - 57.

[6] Surajit C, Rajeev M, Vivek R. Random sampling for histogram construction: how much is enough [A]. In: *Proc ACM SIGMOD* [C]. Seattle, Washington, USA: ACM Press, 1998, **28**: 436 - 447.

[7] Wang Yongli, Xu Hongbing, Dong Yisheng, et al. Design on DSMS supporting distribution system automation [J]. *Automation of Electric Power Systems*, 2004, **28**(13): 85 - 90. (in Chinese)

[8] Arasu A, Manku G. Approximate counts and quantiles over sliding windows [A]. In: *Proc of the* 23*rd ACM SIGACT-SIGMOD-SIGART Symp on Principles of Database Systems* [C]. Paris, France, 2004. 72 - 83.

[9] Gurmeet S, Sridhar R, Bruce G. Approximate medians and other quantiles in one pass and with limited memory [A]. In: *Proc ACM SIGMOD* [C]. Seattle, Washington, USA: ACM Press, 1998, **28**: 426 - 435.

一种基于采样的并行电力负荷数据流划分方法

王永利[1,2] 徐宏炳[1] 董逸生[1] 钱江波[1] 刘学军[1]

([1] 东南大学计算机科学与工程系,南京 210096)
([2] 佳木斯大学计算机公共教研部,佳木斯 154007)

摘要: 为了解决电力工业中并行数据流范围聚集的连续查询问题,提出一种新颖的数据流划分方法. 首先构造了一个适用于数据流处理的扩展蓄水池抽样算法,根据流值变化率引入跳跃因子反应负荷数据的变化情况,实现数据流的自适应并行采样. 然后为了实现数据流量的平均划分,基于近似技术提出2种适应不同情况的生成等深柱状图的算法:增量更新的启发式方法和周期性更新的快捷方法,从而在采样的基础上生成近似划分向量. 通过在实际数据集上对算法性能测试,证明文中提出的数据流划分方法高效实用,适合于高速时变数据流的处理.

关键词: 数据流;连续查询;并行处理;采样;数据划分

中图分类号: TP311